UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LIBARDO ANDREY QUINTERO GONZÁLEZ

Exploring Programmable Networks for Effective Pushback-Based Detection and Mitigation of DDoS Attacks

Thesis presented in partial fulfillment
of the requirements for the degree of
Master of Computer Science

Advisor: Prof. Dr. Luciano Paschoal Gaspary
Coadvisor: Prof. Dr. Alberto Egon Schaeffer Filho

Porto Alegre
September 2022

CIP – CATALOGING-IN-PUBLICATION

González, Libardo Andrey Quintero

Exploring Programmable Networks for Effective Pushback-Based Detection and Mitigation of DDoS Attacks / Libardo Andrey Quintero González. – Porto Alegre: PPGC da UFRGS, 2022.

66 f.: il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2022. Advisor: Luciano Paschoal Gaspary; Coadvisor: Alberto Egon Schaeffer Filho.

1. SDN. 2. Programmable Data Plane. 3. P4. 4. DDoS Attack. 5. Mitigation. 6. Entropy. 7. Pushback. 8. Machine Learning. I. Gaspary, Luciano Paschoal. II. Filho, Alberto Egon Schaeffer. III. Título.

"An investment in knowledge pays the best interest."

— Benjamin Franklin

ACKNOWLEDGEMENTS

First of all, I thank God for all the graces I received. Thanks for giving me strength, health, and amazing people who have been essential throughout my journey.

I thank my grandmother Noris (RIP), who was always by my side, she took care of me, and gave me the honor of welcoming me as her son until the end of her days. I hope I made her proud at all times.

The most sincere thanks to my incomparable mother, Torcoroma, for her love, for always supporting me in everything, helping me overcome all obstacles, and teaching the most significant life lessons.

I would like to thank my amazing uncle, Wilmar, who has been my mentor and best friend, and has accompanied and guided me throughout my life. He has helped and motivated me to be better every day and welcomed me throughout his life as a son.

A special thanks to my beloved wife, Johanna, for the sacrifices we faced together, for being my life partner, supporting, understanding and motivating me at all times. May more achievements be built together in our lives with God's blessings.

I thank my family, family in-law, friends and colleagues for the understanding, who even in the distance always supported me and remained close.

I am immensely grateful to Prof. Dr. Luciano Gaspary and Prof. Dr. Alberto Schaeffer-Filho for the great opportunity and acceptance to have them as advisors. For having believed and invested in me. Thank you for the valuable lessons and teachings. For the guidance, conversations, patience, and understanding that made me a better person.

I am also grateful to Lucas Castanheira and Jonatas Marques for their availability, disposition, and essential help in sharing their research experience and technical expertise to overcome all the presented challenges.

I also thank all the professors and staff of the Informatics Institute of UFRGS for essential teachings transmitted in classrooms and coexistence. The lessons and opportunities that everyone provided me were important for my development and will continue to be in my professional career. I recognize the collaboration of the Computer Networks research group for inspiring academic excellence through its distinguished members.

ABSTRACT

Network infrastructure and servers are targets of different types of attacks daily. One of the most common and devastating types is Distributed Denial of Service (DDoS) attacks, which aim at exhausting resources and directly impacting the availability of services. Although the problem has been investigated for at least two decades, proposals fall short in quickly detecting and mitigating ongoing DDoS attacks while being accurate and pushing the attacks as far as possible from the victim (saving network resources). The emergence of programmable data planes enables novel security solutions with the potential to solve these shortcomings. As a first research effort, in this dissertation, we present BUNGEE, an in-network, collaborative pushback mechanism for DDoS attack mitigation that runs entirely in the data plane. This mechanism is able to, locally at a given switch, identify suspect IP addresses (through the use of continuous IP entropy analysis) and propagate them to other switches. The different switches that are made aware of the suspects enforce a pushback strategy for repelling potential attacks. As an evolution of BUNGEE, we propose BUNGEE-ML, an innovative, hybrid approach that combines the fast processing of the data plane and the high capacity and the intelligence of the control plane for DDoS mitigation. BUNGEE-ML continuously monitors traffic at the data plane to detect network anomalies and supplies machine learning models (running in the control plane) with inputs to perform in-depth traffic analysis. We refer to this as vertical cooperation. Additionally, our approach progressively pushes malicious traffic farther away from the victim through horizontal mitigation coordination between forwarding devices. Our evaluation of a P4-built prototype demonstrates that BUNGEE-ML is highly accurate in identifying and mitigating attack sources due to the vertical cooperation and has a low resource footprint. Furthermore, our pushback strategy saves network bandwidth by mitigating non-legitimate traffic closer to its sources.

Keywords: SDN. Programmable Data Plane. P4. DDoS Attack. Mitigation. Entropy. Pushback. Machine Learning.

Explorando Planos de Dados Programáveis para Detecção e Mitigação de Ataques DDoS Baseadas em Pushback de Tráfego

RESUMO

Infraestrutura de rede e servidores são alvos de diversos tipos de ataques diariamente. Um dos tipos mais comuns e devastadores são os ataques distribuídos de negação de serviço (DDoS - Distributed Denial of Service), que visam esgotar recursos e impactar diretamente na disponibilidade dos serviços. Embora o problema tenha sido investigado há pelo menos duas décadas, as propostas falham em detectar e mitigar rapidamente os ataques DDoS em andamento, ao mesmo tempo em que são precisos e empurram os ataques o mais longe possível da vítima (economizando recursos de rede). O surgimento de planos de dados programáveis permite novas soluções de segurança com potencial para resolver essas deficiências. Como primeiro esforço de pesquisa, nesta dissertação, apresentamos o BUNGEE, um mecanismo de pushback colaborativo em rede para mitigação de ataques DDoS que é executado inteiramente no plano de dados. Esse mecanismo é capaz de, localmente em um determinado switch, identificar endereços IP suspeitos (através do uso de análise contínua de entropia IP) e propagá-los para outros switches. Os diferentes switches que estão cientes dos suspeitos impõem uma estratégia de pushback para repelir ataques potenciais. Como evolução do BUNGEE, propomos o BUNGEE-ML, uma abordagem inovadora e híbrida que combina o rápido processamento do plano de dados e a alta capacidade e inteligência do plano de controle para mitigação de DDoS. O BUNGEE-ML monitora continuamente o tráfego no plano de dados para detectar anomalias na rede e fornece modelos de aprendizado de máquina (executando no plano de controle) com entradas para realizar uma análise de tráfego aprofundada. Nós nos referimos a isso como cooperação vertical. Além disso, nossa abordagem empurra progressivamente o tráfego malicioso para mais longe da vítima por meio da coordenação de mitigação horizontal entre os dispositivos de encaminhamento. Nossa avaliação de um protótipo construído em P4 demonstra que o BUNGEE-ML é altamente preciso na identificação e mitigação de fontes de ataque devido à cooperação vertical e tem um baixo consumo de recursos. Além disso, nossa estratégia de pushback economiza largura de banda da rede, mitigando o tráfego não legítimo mais próximo de suas fontes.

Palavras-chave: SDN, Planos de Dados Programáveis, P4, Ataques DDoS, Mitigação, Entropia, Pushback, Machine Learning.

LIST OF ABBREVIATIONS AND ACRONYMS

SDN	*Software-Defined Networking*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
IP	*Internet Protocol*
DDoS	*Distributed Denial of Service*
PDP	*Programmable Data Plane*
ASIC	*Application-Specific Integrated Circuit*
NPU	*Network Processing Unit*
FPGA	*Field Programmable Gate Arrays*
CPU	*Central Processing Unit*
GPU	*Graphics Processing Unit*
DSP	*Digital Signal Processors*
API	*Application Programming Interface*
ISP	*Internet Service Provider*
IFIP	*International Federation for Information Processing*
IEEE	*Institute of Electrical and Electronics Engineers*
IM	*International Symposium on Integrated Network Management*
TNSM	*Transactions on Network and Service Management*
ML	*Machine Learning*
SVM	*Support Vector Machine*
KNN	*K-Nearest Neighbor*
PCA	*Principal Component Analysis*
CS	*Chi-Square*
RF	*Random Forest*

IG	*Information Gain*
GR	*Gain Ratio*
NB	*Naive Bayes*
MAC	*Media Access Control*
NIC	*Network Interface Controller*
AS	*Autonomous System*
RAM	*Random Access Memory*
TCAM	*Ternary Content-Addressable Memory*
SRAM	*Static Random Access Memory*
BMv2	*Behavioral Model version 2*
FREQ	*Frequency*
CAIDA	*Center for Applied Internet Data Analysis*
CIC	*Canadian Institute for Cybersecurity*
TPR	*True-Positive Rate*
FPR	*False-Positive Rate*

LIST OF FIGURES

Figure 2.1 Traditional network architecture and SDN. 16
Figure 2.2 Top-down approach used in programmable forwarding devices. 17
Figure 2.3 P4 headers definition examples. 18
Figure 2.4 P4 program elements and generic packet processing procedure. 19

Figure 4.1 Overview of BUNGEE considering an attack scenario. 28
Figure 4.2 Operation life cycle of a BUNGEE-enabled switch. 29
Figure 4.3 Pushback process considered in BUNGEE 31
Figure 4.4 Contraction process considered in BUNGEE 32
Figure 4.5 DDoS attack detection and pushback mechanism implemented in the programmable switch. 33
Figure 4.6 Sketch-based data structure for suspect identification. 34
Figure 4.7 Topology used in the experiments. 35
Figure 4.8 BUNGEE accuracy. 37
Figure 4.9 Network utilization savings resulting from BUNGEE. 38

Figure 5.1 Vertical and Horizontal Cooperation. 42
Figure 5.2 BUNGEE-ML's Workflow 43
Figure 5.3 Sketch-based data structure statistics aggregation. 45
Figure 5.4 Statistics aggregation at the end of a monitoring window. 46
Figure 5.5 Packet cloning and transforming into *alarm packet*. 50
Figure 5.6 Pushback process and contraction types. 51
Figure 5.7 Topology used in the experiments. 52
Figure 5.8 Classification metrics for the Machine Learning algorithms adopted in BUNGEE-ML flow classification. 54
Figure 5.9 BUNGEE-ML network utilization savings resulting from the proposed mechanism. 56

LIST OF TABLES

Table 4.1 Reaction time to activate defensive actions. ... 39
Table 4.2 Maximum number of suspects for window size and threshold. ... 40

Table 5.1 Selected features from the CICDDoS2019 dataset. ... 53
Table 5.2 Peak memory for each combination of Feature Selection and ML technique. ... 57
Table 5.3 Wall clock time for each combination of Feature Selection and ML technique. ... 58

CONTENTS

1 INTRODUCTION ... **12**
2 BACKGROUND ... **15**
2.1 Software-Defined Networks and Programmable Networks ... **15**
2.2 P4: Programming Protocol-Independent Packet Processors ... **17**
2.3 Machine Learning Techniques ... **18**
2.4 The *Status-Quo* of DDoS Mitigation ... **20**
3 RELATED WORK ... **22**
3.1 DDoS Detection and Mitigation in the Data Plane ... **22**
3.2 Controller-Based DDoS Detection and Mitigation ... **24**
3.3 Discussion ... **26**
4 BUNGEE: AN ADAPTIVE PUSHBACK MECHANISM FOR DDOS DETECTION AND MITIGATION IN P4 DATA PLANES ... **27**
4.1 Overview ... **27**
4.2 Formalization ... **29**
4.3 Architectural Components ... **32**
4.4 Experimental Evaluation ... **35**
4.4.1 Experiment Setup ... 35
4.4.2 Accuracy ... 37
4.4.3 Network Utilization ... 37
4.4.4 Reaction Time ... 38
4.4.5 Memory Footprint ... 39
5 BUNGEE-ML: A CROSS-PLANE APPROACH FOR A COLLABORATIVE DEFENSE AGAINST DDOS ATTACKS ... **41**
5.1 Overview: Vertical and Horizontal Cooperation ... **41**
5.2 BUNGEE-ML's Workflow ... **43**
5.3 Data Plane Implementation ... **44**
5.3.1 Flow Monitoring ... 44
5.3.2 Statistics-Based Suspect Identification ... 45
5.3.3 DDoS Rapid Response ... 46
5.4 Overcoming Data Plane Limitations ... **47**
5.5 Hybrid Control and Data Plane Operation ... **48**
5.5.1 Flow Classification ... 48
5.5.2 Information Exchange with the Data Plane ... 49
5.6 Pushback Technique ... **49**
5.7 Experimental Evaluation ... **50**
5.7.1 Experimental Setup and Dataset ... 51
5.7.2 Data and Control Plane Cooperation Gains ... 52
5.7.2.1 Accuracy and Precision of Suspects Identification ... 52
5.7.2.2 Network Utilization ... 55
5.7.3 Data and Control Plane Overhead ... 57
5.7.3.1 Data Plane Memory Footprint ... 57
5.7.3.2 Machine Learning Overhead ... 57
6 CONCLUSION ... **59**
REFERENCES ... **60**
APPENDIX A — SUMMARY IN PORTUGUESE ... **65**

1 INTRODUCTION

Distributed Denial of Service (DDoS) attacks are a persistent issue in networked systems. The last few years have shown an increase in both volume and duration of attacks, placing DDoS as one of the leading security concerns in network infrastructures (DONG; ABBAS; JAIN, 2019). Recent reports (*e.g.*, (ATLAS, 2019; PORTER, 2019; KUPREEV O. BADOVSKAYA, 2020)) show DDoS attacks can reach high rates – in the order of Terabits per second – which allows these attacks to overload both servers and network links and ultimately slow down or even halt applications. Overall, attacks on network infrastructures proved to be largely successful in disrupting applications' responsiveness and availability (RANGEL; VESSUM, 2021; DAWSON, 2018).

Over the years, the scientific community has made significant efforts to understand DDoS attacks (BHARDWAJ *et al.*, 2016). As a result of these efforts, several strategies emerged for detecting (DING; SAVI; SIRACUSA, 2021; Kamboj *et al.*, 2017; VARALAKSHMI; THENMOZHI; SASI, 2021; KOUSAR *et al.*, 2021; MACíAS; GASPARY; BOTERO, 2021) and mitigating (Santos da Silva *et al.*, 2016; Agrawal; Tapaswi, 2019; SANGODOYIN *et al.*, 2021) these types of attack. Currently, advances in networking technologies have centered the proposed approaches around two paradigms, namely: (*i*) strategies that operate at the controller in Software-Defined Networks (SDN) or (*ii*) strategies that operate directly in programmable data planes (PDP) (DALMAZO *et al.*, 2021).

In the SDN-controller paradigm (Li *et al.*, 2019; ALSADI *et al.*, 2021; DIMOLIANIS; PAVLIDIS; MAGLARIS, 2021), statistics are analyzed at the control plane to determine the presence of an attack, and then the control plane executes the required actions to overcome the attack (CARVALHO *et al.*, 2020). On the other hand, in the PDP paradigm (Febro; Xiao; Spring, 2019; KALJIC; MARIC; NJEMCEVIC, 2019; NARAYANAN; SANKARAN; SIVALINGAM, 2019), strategies often rely on the P4 language (BOSSHART *et al.*, 2014) for programming DDoS solutions directly in network switches. This allows implementing solutions directly in the data plane without communication with the control plane for decision-making. This provides a significant speedup since it does not involve control plane coordination, and data collection and analysis can be performed at line-rate.

Motivated by the potential mentioned gains of the PDP paradigm, in a first iteration to address the problem of *rapid detection and mitigation* of DDoS attacks, in this dissertation, we propose BUNGEE, a novel fully in-network dynamic attack pushback mechanism. The mechanism is entirely coordinated by the forwarding devices[1], thereby removing the control plane from the critical path and speeding up reactiveness for a prompt defense. In a nutshell, BUNGEE detects a DDoS attack through the entropy analysis of the addresses of incoming packets, since this type of attack is known to cause disturbances in the distribution of source and destination IP addresses. In the proposed pushback mechanism, a forwarding device located next to the victim is expected to alert upstream forwarding devices immediately after entropy disturbances are detected. These devices, in turn, will start to apply a filtering strategy to packets from suspect network flows. Further, these devices will also run the detection mechanism themselves and, if necessary, alert their upstream counterparts, thus effectively "pushing back" the effects of the attack. This process will be repeated in an attempt to confine the attack traffic to as close to the source as possible. Our goal is to ensure the availability of the victim and prevent unnecessary consumption of processing and network resources. As the attack ceases, the mechanism starts to gradually deactivate the filtering procedures in place. Differently from existing data plane approaches for DDoS mitigation that require frequent communication with another network component (*e.g.,* external controller), BUNGEE runs entirely in the data plane, at line rate. This avoids dependence on permanent communication with the controller, which could cause congestion in the communication channel when an attack is in progress (Dargahi *et al.*, 2017).

Despite the promising results of our data plane-based design, we observed that, due to PDP device constraints, it is hard (if at all possible) to devise detection and mitigation mechanisms that are fast and, at the same time, *highly accurate*. Hence, moving further and in an attempt to get the "best of both worlds", we propose BUNGEE-ML, a co-designed control and data plane approach for DDoS mitigation to solve the discussed problems. BUNGEE-ML builds upon our previous work, and we now leverage both the data and control plane to provide a more comprehensive DDoS mitigation mechanism. Our system advocates a multi-level strategy that promotes both vertical and horizontal cooperation between network elements: BUNGEE-ML performs sophisticated traffic analysis outside the switch ASIC as it (*vertically*) relies on resources from the switch CPU and from the SDN controller for *in-depth* traffic analysis; additionally, BUNGEE-ML opportunistically pushes back (*horizontally*) malicious traffic to as far as possible from the victim for *in-breadth* mitigation.

We conducted a comprehensive set of experiments to evaluate both BUNGEE and BUNGEE-ML using real traffic data. Although results for our data plane strategy showed effective mitigation, BUNGEE-ML improved on these results in terms of accuracy by incorporating classifiers into the suspect analysis process, further attenuating the impact of attack flows on network

[1]We use the terms (*forwarding*) *device(s)* and *switch(es)* interchangeably throughout the document.

resources and reducing the penalization of legitimate flows.

Main Research Contributions. In summary, we make the following contributions:

- We devise a *network-wide communication strategy* for switch coordination using P4 data plane primitives.
- We design a *control-plane-free pushback mechanism*, which builds upon our communication strategy to implement a “quick-mounting defense barrier” against DDoS.
- We integrate an efficient *IP entropy anomaly detection approach* to the pushback mechanism to “make the barrier a moving one”, avoiding wasting network resources.
- Based on our experience with BUNGEE, we design a hybrid approach that combines control and data planes to mitigate network-wide DDoS attacks.
- We implement an ML-powered classifier that enhances the accuracy of our in-network DDoS detection while maintaining the detector’s ability to execute at line-rate.
- We present extensive experiments that demonstrate the applicability and effectiveness of our proposed approaches.

The remainder of this work is structured as follows. In Chapter 2, we present the background on SDN, programmable networks, P4 language, machine learning and DDoS attack mitigation. In Chapter 3, we discuss the state-of-the-art in DDoS attack detection and mitigation using programmable data planes and intelligent solutions using control plane controllers. In Chapter 4, we introduce BUNGEE, our collaborative pushback mechanism for DDoS attack mitigation that runs entirely in the data plane. In Chapter 5, we introduce BUNGEE-ML, our novel hybrid approach that combines the fast processing of the data plane and the high capacity and intelligence of the control plane for accurate attack detection and mitigation in the network. Finally, in Chapter 6, we present our concluding remarks and perspectives for future work.

2 BACKGROUND

In this chapter, the theoretical concepts that served as the basis for the development of this work are presented. In Section 2.1, we briefly introduce the fundamental concepts of Software-Defined Networking and data plane programmability. In Section 2.2, we describe the P4 language for programming packet forwarding devices. In Section 2.3, we review fundamentals of machine learning techniques that can be considered in in-network security designs. Finally, in Section 2.4, we review strategies that employ SDN for attack mitigation.

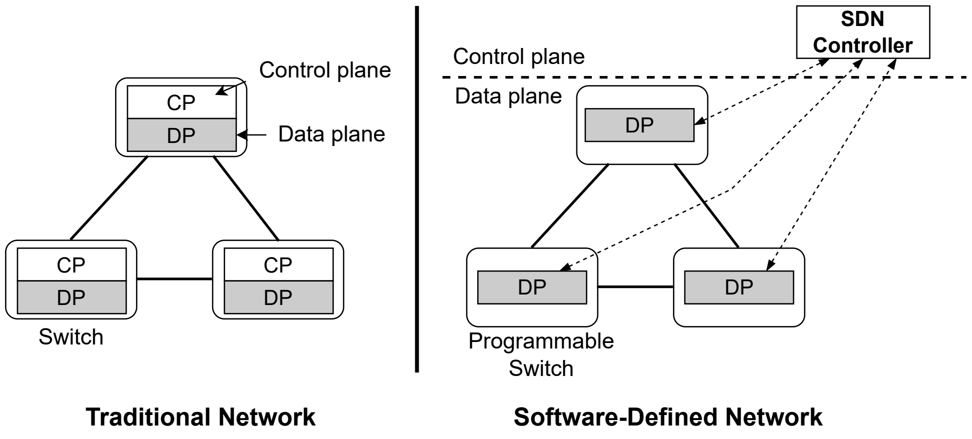
2.1 Software-Defined Networks and Programmable Networks

The rapid growth of existing networks and the continuous demand for new applications with ever-increasing performance have increased the complexity of traditional networks. Traditional networks include forwarding devices (switches and routers) of different capacities, technologies, and vendors. Network operators implement control policies and configure devices individually, limiting a global view of the network state and ultimately hampering network management as the network grows. In addition, forwarding devices are limited to preset manufacturers' functions and features, making it difficult for operators to implement their algorithms (by the restriction of internal device structures) and turning this network model highly dependent on vendor support (FEAMSTER; REXFORD; ZEGURA, 2014).

Software-Defined Networking (SDN) changed the way networks are designed and managed, introducing the network programmability to promote innovation in network management and reducing the barrier to deploying new services in the network (XIA *et al.*, 2015). The first research efforts in programmable networks emerged in the early to mid-1990s. The rise of the Internet and support for diverse network applications motivated researchers to propose new ideas to improve network services (FEAMSTER; REXFORD; ZEGURA, 2014). At that time, active networks were proposed to solve several issues in network technologies identified by the research community. They envisioned an Application Programming Interface (API) to expose forwarding device resources (*e.g.*, processing, packet queuing, storage), similar to how programming languages disclose server resources. A network operator could use this API to develop new approaches to define the processing of data packets (TENNENHOUSE *et al.*, 1997). Despite considerable research efforts, there was no sufficiently compelling (or attractive) solution or clear implementation path that would drive the widespread deployment of this technology. The unstoppable growth of networks and the high volumes of traffic turns the control and management of the network into a complex task, which has pushed forward the adoption of the SDN paradigm (FEAMSTER; REXFORD; ZEGURA, 2014).

SDN is a dynamic network architecture that seeks to help and accelerate the implementation of new services and respond quickly to changes in the demand for requirements (MCKEOWN *et al.*, 2008). As Figure 2.1 illustrates, different from traditional networks, SDN proposes an architecture in which the operations (data plane) and network control (control plane) are decoupled. The latter, removed from the network devices, becomes centralized in an external entity (MCKEOWN *et al.*, 2008; Kreutz *et al.*, 2015).

Figure 2.1 – Traditional network architecture and SDN.



Source: Adapted from (DUNGAY, 2019)

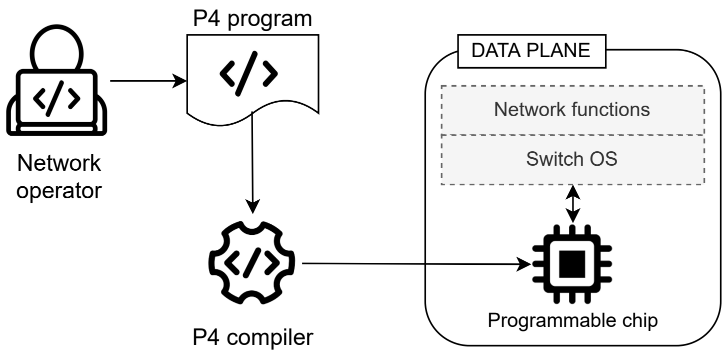
The *Data plane* is comprised of interconnected forwarding devices that perform a set of elementary operations (*e.g.,* forward to specific ports, drop, forward to the controller, rewrite some header) used to take actions on the incoming packets. On the other hand, the *Control plane* refers to a logically centralized controller that, through an open interface such as the OpenFlow protocol (MCKEOWN *et al.*, 2008), becomes aware of all network elements and their status (*i.e.*, it has a global network view). The controller handles decision-making about the flows and, as it can interact directly with switches, updates the forwarding rules directly in the data plane.

A new paradigm has recently promoted further progress in SDN networks: programmable data planes (PDP). Data plane programmability enables the development of new functions to be deployed directly in forwarding devices, including security functions (DALMAZO *et al.*, 2021; CARVALHO *et al.*, 2021; KFOURY; CRICHIGNO; BOU-HARB, 2021; HAUSER *et al.*, 2021; PAOLUCCI *et al.*, 2019). Although a programmable switch has fewer resources than a control plane controller, it can still execute more than just forwarding logic. The advantage of running applications in the data plane is they run at a line rate, increasing reactiveness without depending on the control plane for decision making.

2.2 P4: Programming Protocol-Independent Packet Processors

Data plane programmability provides network operators with the ability to access forwarding devices and reprogram them to, for example, implement new or customized protocols and run innovative services. P4 (BOSSHART *et al.*, 2014) has emerged as a high-level language for programming the data plane. It was designed to define packet processing pipelines and describe the functions executed by a programmable forwarding device. P4 is suitable for specifying the behavior of various device types (*e.g.,* ASICs, NPUs, FPGAs, and software switches), which makes it target-independent. Like other units that execute code written in a specific programming language (*e.g.,* CPU, GPU, or DSP), a "top-down" approach is used in programmable forwarding devices. The network operator defines the feature set of the network in the P4 program, and then the code is compiled and loaded into the processor. Figure 2.2 illustrates the concept.

Figure 2.2 – Top-down approach used in programmable forwarding devices.



Source: The authors (2022)

A P4 program begins with a data declaration section to define the packet header format and metadata information, describing the structure, the sequence, and the length of a series of fields. The user-defined data structures associated with each packet are known as *user-defined metadata*. In contrast, the metadata provided by the architecture associated with each packet (*e.g.*, timestamp, and the input port from where a packet has been received) is known as *intrinsic metadata.* To parse a packet, it is required to declare, in the form of (standard and custom) protocols, all the headers that the programmable forwarding device can process. As an example, Figure 2.3 illustrates the declaration of the standard Ethernet header and a custom protocol. Finally, data declaration is mapped to a *header and metadata bus* that maintains this information available through all processing stages.

Figure 2.3 – P4 headers definition examples.

```
header ethernet_h {
        bit<48> src_addr;
        bit<48> dst_addr;
        bit<16> ether_type;
}
```

```
header custom_h {
        bit<48> src_addr;
        bit<16> pkt_num;
        bit<16> protocol;
}
```

Source: The authors (2022)

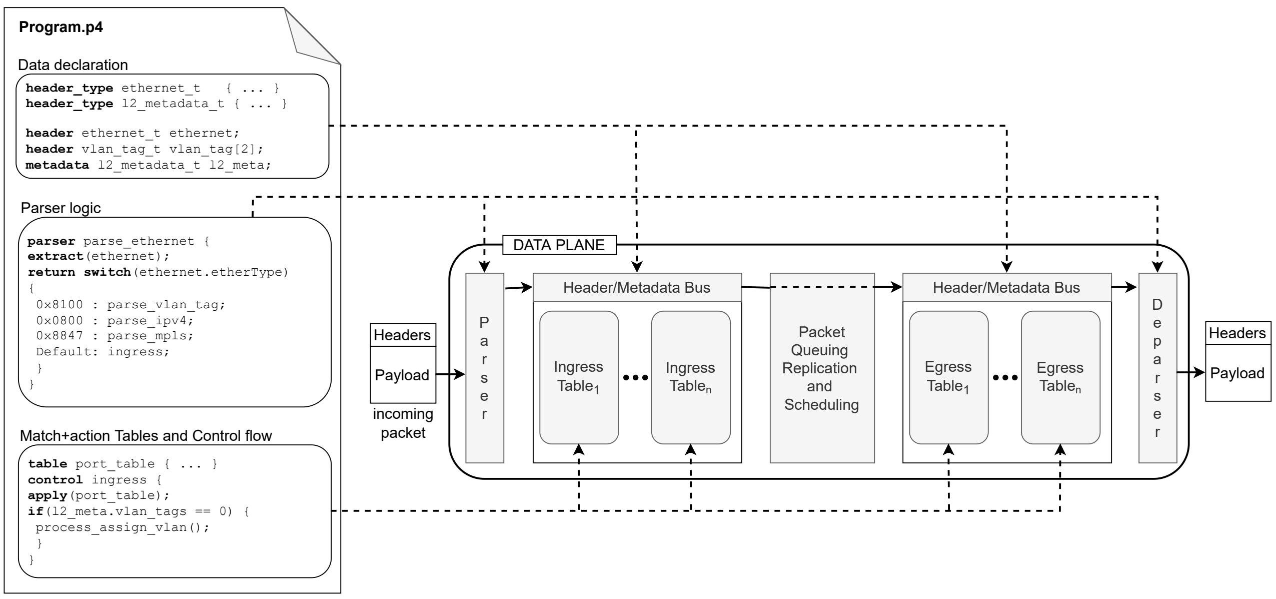
A P4 program also defines a parser, which determines how to handle incoming packets. This parser supports the processing of custom headers, enabling the design of novel protocols. In the parser, the fields of all defined headers are extracted, and the packet processor pipeline continues to *match+action* tables. These tables describe what actions may be applied to packets when header fields match the tables. Some examples of actions that these tables may take are modifying packet headers, modifying metadata, cloning packets, recirculating packets, as well as choosing the output port for packets, or determining that they should be dropped. Finally, a deparser rebuilds the packet and appends the necessary headers, at which point the switch can transmit the packet. Figure 2.4 illustrates a generic packet processing procedure as followed by a P4 program.

2.3 Machine Learning Techniques

Machine learning (ML) is a data analytics technique that enables a system to learn from experience. ML uses algorithms to learn information through large-scale observations. ML techniques are grouped according to supervised learning, unsupervised learning, and reinforcement learning (DALAL, 2020):

- *Supervised Learning*: infers a model from a labelled dataset and its features (*i.e.,* individual properties or characteristics). This dataset is used to train the ML algorithm and develop a regression or classification model. So, the technique requires a suitable and useful feature set for efficiently building the model that will serve as the basis for classification.
- *Unsupervised Learning*: tries to adjust the model according to the observations. This

Figure 2.4 – P4 program elements and generic packet processing procedure.



```
header_type ethernet_t   { ... }
header_type l2_metadata_t { ... }

header ethernet_t ethernet;
header vlan_tag_t vlan_tag[2];
metadata l2_metadata_t l2_meta;
```

```
parser parse_ethernet {
extract(ethernet);
return switch(ethernet.etherType)
{
 0x8100 : parse_vlan_tag;
 0x0800 : parse_ipv4;
 0x8847 : parse_mpls;
 Default: ingress;
 }
}
```

```
table port_table { ... }
control ingress {
apply(port_table);
if(l2_meta.vlan_tags == 0) {
 process_assign_vlan();
 }
}
```

Source: Adapted from (CORDEÍRO; MARQUES; GASPARY, 2017)

is performed by processing the input samples as random variables and discovering similarities within unlabelled information for grouping it into classes. So, unsupervised is distinguished from supervised learning because there is no previous knowledge of features for classifying.

- *Reinforcement Learning*: in reinforcement learning, an agent attempts to learn an optimal policy that maximizes a given “reward function”. The agent executes actions in the environment, which moves from one state to another. The agent receives a positive feedback if the new state is considered better than the previous one, or a negative feedback otherwise. The goal of the agent is to maximize the expected cumulative reward.

In the context of network security, ML techniques can be used to assist in the detection and mitigation of malicious traffic (MUSUMECI *et al.*, 2020). However, they commonly run on a dedicated server or controller due to the complex operations required. Supervised techniques are usually employed for the detection of attacks since one wants to classify the flow traffic in one of two classes already defined (*e.g.,* benign or malign) and not to discover new classes (ARSHI; NASREEN; MADHAVI, 2020; WEHBI *et al.*, 2019).

Among some representative examples of supervised techniques used for classification, *Naive Bayes* is a classification approach based on the principle of conditional class independence of the Bayes theorem. The algorithm builds on the fact that the presence of one feature does not affect the presence of another on the probability of a given outcome. Each predictor has the same effect on the outcome. The algorithm performs a probabilistic classification of observations, grouping them into predefined classes. *Support Vector Machine (SVM)* constructs a hyperplane (a line separating a set of input data) that best differentiates two classes (*i.e.,* where the distance between classes of data points is at its maximum). The points located close to this line are called support vectors. SVM builds a model that assigns new entries to one category based on which side of the space they are placed. *K-Nearest Neighbor (KNN)* classifies data points based on their proximity and association with other available data. This means that the algorithm calculates the distance between the data points and then gives a value to a new sample based on how close it is to the data points in the training set. As a final example, *Random Forest* constructs several decision trees (a tree-like model of decisions and their possible consequences) from the existing dataset features. Then, the trees are merged to reduce variance and classify the given values.

A crucial aspect of efficient machine learning models is feature selection. This preprocessing task involves identifying and selecting the relevant features in the original dataset. This process eliminates as much information as possible that is irrelevant and redundant for the classification process. The result is a subset of features that will serve as the input for the machine learning techniques presented above. This subset is an enhanced version of the original set since it has reduced data size (and, therefore, computational complexity). Several algorithms can be used for feature selection, including Principal Component Analysis (PCA), Information Gain, Gain Ratio, and Chi-Square, among others (ILLAVARASON; SUNDARAM, 2019; WANG; XIAO; RAJASEKARAN, 2020).

2.4 The *Status-Quo* of DDoS Mitigation

The possibility of developing mitigation mechanisms to be deployed in the network has allowed it to be reactive and defend itself from attacks without the intervention of a network operator. These mitigation mechanisms are focused on two paradigms: the SDN-controller paradigm and the PDP paradigm.

In the *SDN-controller paradigm*, the controller is provided with traffic statistics collected in the data plane. The controller uses this data to analyze the network behavior and identify anomalies that might indicate an attack is happening. In addition, the controller may execute mitigation techniques to halt the attack, which involves reconfiguring the forwarding devices (RAGHU-

NATH; KRISHNAN, 2018). Next, we briefly describe three techniques typically used in the SDN-controller paradigm.

- *Pushback*: is a collaborative defense strategy in which the main idea is to begin attack mitigation at devices closest to the victim and expand the defenses to its upstream neighbors (*i.e.,* in the opposite direction to the attack). In a pushback strategy, the controller configures rules on forwarding devices as needed to mitigate the attack until it identifies that the attack has stopped.
- *Throttling*: is a strategy based on rate limiting. The controller installs rules that introduce additional delays on forwarding devices after an attack is detected. Rules are only installed on devices in the path to the victim or all devices directly connected to the victim. This choice depends on how the strategy is configured. This technique differs from pushback, where all upstream devices participate in the mitigation.
- *Honeypot*: works as a "trap" for the attackers and their malicious traffic. Once the attack is identified, the main idea is to redirect all the malicious traffic to a different destination than the victim. Unlike previous techniques, this traffic redirection rule is usually installed on the network's edge. The victim and the entire network are then isolated from the attack. At the same time, the attacker is deceived into believing that his attack is successful.

In the *PDP paradigm*, the data plane is independent and executes by itself all the necessary actions. Four main strategies can be used to reduce a DDoS attack on a network when this paradigm is adopted (BAWANY; SHAMSI; SALAH, 2017). These strategies are implemented as a rule, which is applied based on the packet's source.

- *Drop*: all packets coming from addresses identified as sources of the ongoing attack are discarded. This technique is quite aggressive and requires a high degree of certainty in the classification of attack sources.
- *Filter*: just a fraction of traffic thought to be malicious is discarded. The advantage of this approach is that incorrectly identified traffic (*i.e.,* legitimate packets regarded as malicious) will not be blocked entirely.
- *Redirection*: consists of forwarding suspicious traffic to a special device on the network for further inspection. While it is a compelling approach, it imposes non-negligible overheads regarding forwarding device processing and network bandwidth.
- *Quarantine or Traffic Isolation*: malicious traffic is (temporarily) confined to prevent network resources from being overwhelmed by the attack. It might be impractical in specific scenarios because it demands large amounts of storage resources.

3 RELATED WORK

This chapter discusses the recent literature related to DDoS detection and mitigation challenges. We start by describing approaches that benefit from programmable data planes in Section 3.1. Then, in Section 3.2, we review the literature that enlists control plane devices to execute detection and mitigation actions. Finally, in Section 3.3, we discuss the presented approaches and strategies, as well as the limitations in relation to the proposal of this work.

3.1 DDoS Detection and Mitigation in the Data Plane

Next, we present innovative research efforts using only programmable data planes to handle attacks. Zhang *et al.* (ZHANG *et al.*, 2020) introduce Poseidon, which serves as an orchestrator for grouping a set of DDoS mitigation mechanisms (*i.e.*, security policies) in the data plane and implementing them according to the nature of the attack. Poseidon leverages programmable data planes to provide a modular and straightforward language where operators can express a range of security policies to defend against a diverse set of DDoS attacks.

Febro *et al.* (Febro; Xiao; Spring, 2019) propose an approach in which forwarding devices in the network edge are programmed to keep a per-port record of the number of packets received per second. When this number is higher than a configured threshold in a port, the forwarding device automatically drops subsequent connection requests in that specific port. The dropping remains until no attack is detected for an operator-determined interval. This work presents a contribution in exploring the available P4 primitives. However, its simplified design penalizes all incoming requests, even legitimate ones, since blocking is port based (as opposed to flow or source based, for example).

Kaljic *et al.* (KALJIC; MARIC; NJEMCEVIC, 2019) envision the implementation of a firewall in the data plane. The proposed solution can detect an attack by monitoring the changes in the counter values of traffic flows. When a proportional increment in the number of incoming packets and the number of destination unreachable responses is perceived, the switch assumes an attack has been detected. The proposed solution handles the attack in two possible ways. The first is configuring rules for dropping attack flows on an input/output interface. A second way is redirecting flows to a honeypot. Despite the gains of the solution, the detection mechanism limits its general applicability, as considering only unreachable destination messages can lead to incorrect attack detection, *e.g.*, when the communication with the destination application fails for different reasons. Similarly, it is unclear how the incoming flows are confirmed to be malicious, which is crucial in order to allow legitimate requests to still have access to the desired services (when the network is under attack).

Narayanan *et al.* (NARAYANAN; SANKARAN; SIVALINGAM, 2019) present a P4-based framework to safeguard a device connected to a programmable switch from being used to perform an IP address spoofed attack. For this purpose, the proposed solution keeps a MAC-based listing of the devices allowed to connect with the switch. Using this information, the switch builds a ‘whitelist’ with only the assigned IP addresses to devices with established connections. All packets arriving at the switch will be subject to an IP source inspection. Packets with source IP included in the whitelist will be processed. Otherwise, they will be dropped. The work proposes a promising solution to ensure a programmable switch is used only for legitimate actions. However, it introduces the need to manually update the whitelist whenever new devices join the network or require an IP address change; otherwise, legitimate flows may be incorrectly discarded.

Friday *et al.* (FRIDAY *et al.*, 2020) propose a strategy to address DDoS attack vectors. The work performs detection based on a threshold of requests in a specific window. If the threshold is surpassed, the switch considers executing two mitigations actions according to the characteristics of the requests. The first action is to drop subsequent requests on the switch itself and thus slow down the attack. Another action is dropping requests at the network edge, thus preventing the attack from spreading within the network. Despite the favorable results obtained, the implemented strategy requires some operations to be executed in a controller. Although mitigation does not depend entirely on these actions, they play an important role in minimizing the penalty over legitimate requests by adjusting the detection sensibility according to the network condition.

Similarly, Nadim *et al.* (NADIM; FOYSAL, 2021) introduce an approach to detect as well as mitigate UDP flood attacks using a programmable data plane. For the detection, they employ the entropy measurement of incoming source IP addresses on each switch port. In the presence of an attack, the entropy is disturbed and, eventually, a threshold is exceeded. This causes the blocking of the port that produced the disturbance. Although this work is effective in reducing the impact of attacks, it can penalize a high number of legitimate packets. This is due to the implementation of such a coarse mitigation action as port blocking.

Yaegashi *et al.* (YAEGASHI; HISANO; NAKAYAMA, 2021) propose a strategy to detect and subsequently mitigate flooding attacks using programmable devices located at the edge of the network. The detection is made by monitoring the rate at which a switch receives flows and assigns them to available queues. When this rate exceeds a threshold, the switch assumes an attack is in progress. More specifically, the programmable switch identifies the overloaded queue and reallocates flows that arrive to release the compromised queue. Finally, all the packets belonging to reallocated flows are discarded. The proposed strategy impacts the forwarding of legitimate flows by establishing a static threshold and dropping them indiscriminately. This design decision does not take into account that the arrival rate of legitimate flows can increase at some point due to reasons other than attacks, such as, for example, a sudden growth in service popularity.

Finally, Ilha *et al.* (ILHA *et al.*, 2020) introduce EUCLID, a purely data-plane-based mechanism for detecting and mitigating volumetric DDoS attacks. In EUCLID, traffic flows are classified as legitimate or malicious according to a threshold on the variation between the frequency of their IP addresses before and after an attack has been detected. The work represents a promising advance in the timely detection and mitigation of DDoS attacks. However, the strict time and memory constraints imposed by current programmable hardware limit the accuracy of flow classification. As a result, EUCLID requires careful parameterization for the expected frequency variations to be able to accurately discriminate between malicious and legitimate flows.

3.2 Controller-Based DDoS Detection and Mitigation

In contrast to data plane-only approaches, strategies centered around the control plane are way more common as a result of more than a decade of research on software-defined networking. Li *et al.* (Li *et al.*, 2019) implement a whitelist table that can be used for filtering off spoofed IPs. This table is formed by legitimate IP addresses that have previously established TCP connections with targets of interest. Due to the complexity and required resources (*e.g.*, memory) to keep track of each connection state, the creation of such a whitelist is performed on the control plane. To speed up the analysis of incoming packets, a shadow version of the whitelist is maintained on the data plane (and updated periodically by the external SDN controller). Hence, packets from IPs in the data plane whitelist are forwarded to their destination without further processing. Otherwise, they are sent to the control plane for analysis. The main downside of this approach is the high communication overhead between control and data planes to synchronize the whitelist and deal with new flows.

Dimolianis *et al.* (DIMOLIANIS; PAVLIDIS; MAGLARIS, 2021) propose a signature-based traffic mitigation architecture implemented using machine learning models in the control plane to deal with DDoS amplification attacks. The work monitors network traffic in the data plane and produces signatures considering several pre-selected fields extracted from packet headers. Next, signatures are sent to the control plane to categorize them as benign or malicious using machine learning methods. The signatures are returned to another component in the data plane, where all packets directed to destinations of interest are examined. Malicious packets are dropped, while benign packets are returned to the pipeline to be forwarded to the destination. Despite showing effective attack mitigation, this approach slows down the forwarding process by subjecting all packets towards destinations of interest to additional analysis. In addition, filtering packets based on signatures gives room for an attacker to change the characteristics of the packets so as not to be dropped.

The approaches described so far consider switches in a network independently. Next, we discuss approaches that involve the cooperation of multiple devices in the network to mitigate attacks. Bülbül and Fischer (Bülbül; Fischer, 2020) introduce a mitigation framework that uses pushback to configure OpenFlow filtering rules on devices belonging to the victim network. However, pushback actions are only carried out by the controller when a device capacity is already exceeded, with the same rule set being broadcasted indiscriminately to all upstream switches.

Zhang *et al.* (Zhang *et al.*, 2019) carry out an exhaustive analysis of the operation and gains of the pushback technique. Thus, they propose a framework designed to assist the deployment of defenses at strategic locations in the path to a DDoS attack victim. The proposed approach employs a controller for ranking strategic sites in which to activate defenses. This ranking is based on two criteria. The first is the collected information on the device about forwarding attack traffic to the victim. The second is the topological place the device is located. Because this solution considers the amount of attack traffic on devices to deploy defenses, the deployment can be uncontrolled since an attacker could flood the entire network with the attack and force the controller to evaluate new devices each time.

Mi *et al.* (MI; WANG, 2019) introduce ML-Pushback. Based on the information collected on a victim device, this framework employs machine learning techniques in a controller to infer traffic rate thresholds. The controller configures these thresholds on the upstream switches to perform traffic throttling on attack traffic. With such simplistic traffic limitations rules installed on devices, this framework could penalize a large number of legitimate flows when an attack is detected.

Hammed et al. (HAMEED; KHAN, 2018) leverage SDN to introduce a collaborative mitigation strategy based on pushback. The strategy contemplates the communication of controllers lying in different autonomous systems (AS). When a network is under attack, the controller informs the neighboring controllers to block the attack. Hence, the effect of the attack is limited along the attack path before arriving at the victim. This strategy involves collaboration between ASes, which increases complexity. Otherwise, the solution is limited to being deployable in controllers belonging to the same network.

3.3 Discussion

Approaches that focus on the data plane (Febro; Xiao; Spring, 2019; KALJIC; MARIC; NJEMCEVIC, 2019; NARAYANAN; SANKARAN; SIVALINGAM, 2019; FRIDAY *et al.*, 2020; ZHANG *et al.*, 2020; YAEGASHI; HISANO; NAKAYAMA, 2021) can react more quickly to attacks but lack accurate techniques to identify the malicious traffic. This limits the effectiveness of these approaches when trying to mitigate attacks. In contrast, approaches deployed in the control plane (HAMEED; KHAN, 2018; Li *et al.*, 2019; Zhang *et al.*, 2019; MI; WANG, 2019; Bülbül; Fischer, 2020; DIMOLIANIS; PAVLIDIS; MAGLARIS, 2021) can be more sophisticated and accurate but introduce a delay in reaction, substantial overhead on network resources, and the required communication between planes as a new point of failure.

In a first effort to address the mentioned issues, we devised a pushback-based mechanism for DDoS attack mitigation that runs entirely in the data plane. The proposal, entitled BUNGEE, is described next in Chapter 4. As we show next, the overall results achievable are very good, especially concerning rapid response to an ongoing attack. Nevertheless, given the limitations of data plane devices, we identified that attack detection, *i.e.*, the confirmation of suspect sources as actual attackers, would benefit if more refined classification mechanisms were employed. This led us to conduct a second research effort and propose BUNGEE-ML. This enhanced system executes a lightweight technique in the data plane for quick attack reaction and employs machine learning models in the control plane as a further, more accurate step into convicting malicious flows and absolving legitimate traffic. Additionally, our work deals with the coordination between many switches in the network (or across networks), enabling the mitigation to converge on the culprit sources of attack. BUNGEE-ML is presented in Chapter 5.

4 BUNGEE: AN ADAPTIVE PUSHBACK MECHANISM FOR DDOS DETECTION AND MITIGATION IN P4 DATA PLANES

In this chapter, we introduce BUNGEE, our in-network, collaborative pushback mechanism for DDoS attack mitigation[1]. In Section 4.1, we present an overview of its operation. In Section 4.2, we describe its foundations, and in Section 4.3, we detail its main components, their interactions, and how they are instantiated in P4. Finally, in Section 4.4 we describe the experimental setup and the obtained results.

4.1 Overview

BUNGEE[2] was designed to deal with volumetric DDoS attacks, where, in a coordinated way, multiple sources send a large amount of forged traffic to targets, depleting their processing and network resources. Figure 4.1 shows a network attack scenario and the general idea behind BUNGEE. It illustrates three different switch "levels" taken into account in our design. The first level refers to the *victim's* switch. It is where the victim of the attack is directly linked to and receives all attack traffic sent to it. Next, *intermediate* switches belong to the next level. These are forwarding devices located either on the same AS or any AS connecting the victim network to the attack source networks. Finally, *edge* switches are equipment to which the attack sources are immediately connected to.

BUNGEE supports *two operation modes*:

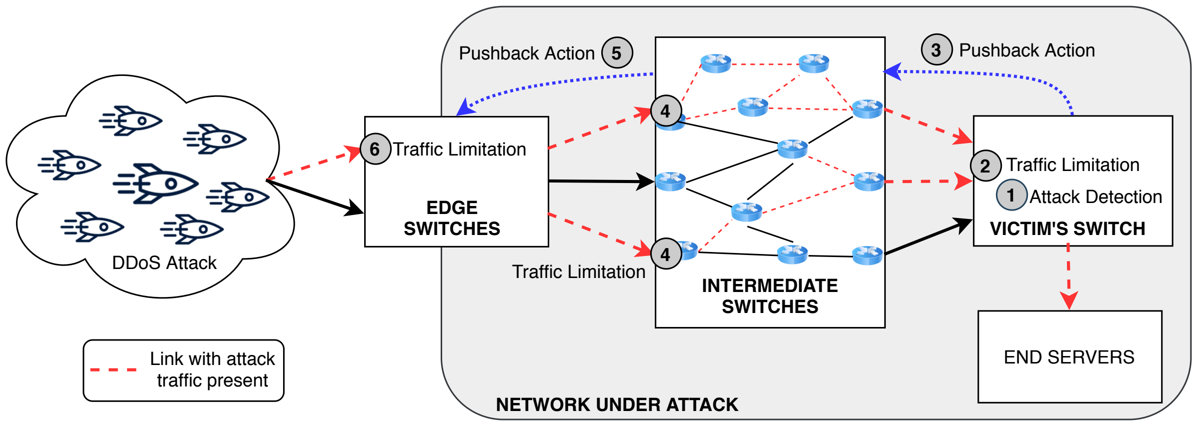
- *Intra-AS, corporate deployment:* subnetworks are spread across a metropolitan or regional area but under the management of a single administrative domain, facilitating deployment. In this scenario, it is possible to pushback a DDoS attack to as far as the domain border device.
- *Inter-AS deployment:* switch coordination can span over multiple cooperating AS's across the Internet. In this scenario, mutual agreements (like those required for peering) are necessary, but the pushback can go to as far as the edge switches of the source network(s)

[1]This chapter is based on the following publication:

- Libardo Andrey Quintero González, Lucas Castanheira, Jonatas Adilson Marques, Alberto Schaeffer-Filho, Luciano Paschoal Gaspary. **BUNGEE: An Adaptive Pushback Mechanism for DDoS Detection and Mitigation in P4 Data Planes.** 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM 2021)(GONZáLEZ *et al.*, 2021).

[2]BUNGEE is named after *Bungee Jumping*, a sports activity that abstractly resembles our mechanism, in which a person leaps down from a high height, connected to an elastic cord. The jumper oscillates up and down until energy vanishes and stability is reached.

Figure 4.1 – Overview of BUNGEE considering an attack scenario.



Source: The authors (2021)

of an attack.

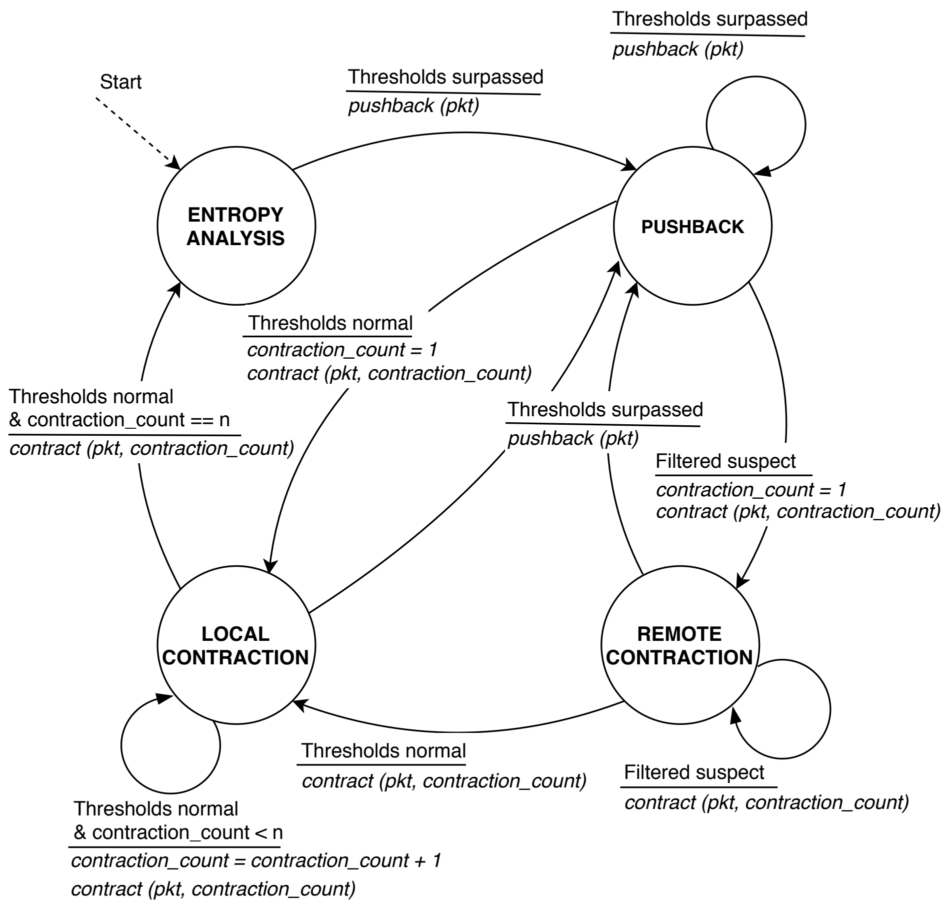
The first step of BUNGEE is – once a DDoS attack is detected (which will be described in the following subsection) on the victim's switch (Label 1 in Figure 4.1) – determining a set of IP addresses as candidate attack sources and starting to filter them locally (2). At this time, the neighbor intermediate switches are alerted about the ongoing attack to take the necessary actions, *i.e.,* a *pushback* action is executed (3). This communication is carried out through *alarm* packets containing a list of suspect IP addresses. Right after a neighbor switch receives an alarm, it starts its own detection process. If it detects an ongoing attack, it starts filtering packets from informed suspects (4). Additionally, it notifies its upstream switches (5), which re-iterate through the described steps. Ideally, the pushback mechanism is expected to end at the edge switches (6). In this case, malicious packets will be discarded as close as possible to their sources, thereby avoiding network resource exhaustion.

Although not shown in the figure, an intermediate/edge switch that starts filtering packets for suspect IP addresses also sends a *notification* message (containing IP addresses that the switch has started filtering) to its downstream switches. If a switch receives notifications from all of its upstream neighbors, it means it has successfully handed off filtering of an IP address to its upstream counterparts, and so it stops filtering packets from that IP, making room for new entries. Finally, as we will detail ahead, the proposed solution is adaptive to fluctuating attack dynamics. For example, when an ongoing attack ceases, our mechanism starts a “contraction” process, where filtering actions are gradually deactivated.

4.2 Formalization

The finite-state machine (FSM) in Figure 4.2 formalizes the states in which a switch running BUNGEE may be at any point in its operation life-cycle. We focus here on the inter-switch communication processes. In Section 4.3, we will detail the fine-grained operation steps of the proposed mechanism.

Figure 4.2 – Operation life cycle of a BUNGEE-enabled switch.



Start
ENTROPY ANALYSIS
PUSHBACK
LOCAL CONTRACTION
REMOTE CONTRACTION
Thresholds surpassed / pushback (pkt)
Thresholds surpassed / pushback (pkt)
Thresholds normal / contraction_count = 1 / contract (pkt, contraction_count)
Thresholds normal & contraction_count == n / contract (pkt, contraction_count)
Thresholds surpassed / pushback (pkt)
Filtered suspect / contraction_count = 1 / contract (pkt, contraction_count)
Thresholds normal / contract (pkt, contraction_count)
Filtered suspect / contract (pkt, contraction_count)
Thresholds normal & contraction_count < n / contraction_count = contraction_count + 1 / contract (pkt, contraction_count)

Source: The authors (2021)

A forwarding device can detect disturbances in the network traffic via entropy analysis, as discussed below. After detecting an entropy alteration, a pushback action is executed to notify all upstream switches. As a monitoring window ends, the device continues calculating entropy for the following windows. After a no entropy disturbance window, the switch falls into a local contraction state, indicating that the attack is being contained. However, if an entropy disturbance is registered for a window while in this state, the switch performs a pushback action again. To avoid oscillating activation/deactivation of defensive measures, the switch waits for some entropy normality windows to inactivate the defenses (local contraction)[3]. Next, we discuss in detail the main aspects of this FSM.

IP Entropy Analysis. Before the pushback action can kickoff, BUNGEE needs first to detect an ongoing attack. To this end, we employ *IP entropy analysis* in the data plane, building on our research group previous work (Lapolli; Adilson Marques; Gaspary, 2019). In information theory, the entropy of a variable is the level of information inherent in the variable's possible outcomes. It is appropriate to analyze the behavior of network indicators due to its high sensitivity in detecting abnormal network traffic changes. We make use of *source* and *destination IP entropy analysis* for attack detection.

A volumetric DDoS attack causes disturbances in the distribution of source and destination addresses. During a DDoS, the entropy of source addresses tends to increase due to the wide distribution of the attacking (potentially forged) IP addresses. Conversely, the entropy of destination addresses tends to decrease due to the high recurrence of the victim IP address. For discretization purposes, we divide incoming packets into windows of a fixed number of packets. In this context, Shannon's entropy offers the degree of randomness of a monitoring window *(X)*, and can be calculated, per source (H_{src}*(X)*) or destination (H_{dst}*(X)*) IP, as follows:

$$H(X) = \log_2(m) - \frac{1}{m}\sum_{x=0}^{N} f_x \log_2(f_x) \tag{4.1}$$

In Equation 4.1, *m* corresponds to the number of packets in a window, and $f_0, f_1, \ldots, f_N$ represent the frequencies of each (source or destination) address in the window *X*. In case of a DDoS attack, the destination IP entropy tends to zero, $H_{dst}(X) \to 0$. In contrast, the source IP entropy tends to the maximum value, $H_{src}(X) \to MAX$. The entropy values calculated for each monitoring window are compared with thresholds set according to entropy measurements observed during the "normal" network operation. If these values are exceeded, BUNGEE will start the pushback process, as described below.

Pushback Process. The pushback process deploys defenses progressively at the upstream devices, shifting to as far as possible from the victim network. After an entropy disturbance, the victim's switch will alert its first-hop upstream counterparts, sending them an *alarm packet* containing a list of suspect IP addresses. Upon receiving an alarm, upstream neighbors will begin calculating the entropy for the upcoming windows and filtering packets incoming from

[3]Note that state changes occur at different time frames since the attack is detected. It is identified in the monitoring window W_t. First-hop upstream switches perform entropy analysis in the next time window W_{t+1}. Consequently, edge switches will be warned in window W_{t+m}, where m corresponds to the number of intermediate switches.

the suspect IPs.

Usually, the controller executes pushback by sending a particular packet to the switches. Due to limitations in programmable hardware, spontaneous packet generation is not possible. To address this limitation, we resort to the P4 `clone` primitive (CONSORTIUM, 2020), which allows making a copy of a packet. Thus, pushback is executed by cloning the last packet completing the monitoring window, and using it for inter-switch communication. The cloned packet is transformed into an *alarm packet* by appending a custom header and the identified suspect IPs. Finally, the alarm packet is sent to upstream devices. Algorithm 4.3 summarizes the described pushback operation.

Figure 4.3 – Pushback process considered in BUNGEE

```
    Input: Incoming packet completing monitoring window (pkt)
createAlarm(pkt);
SuspectIPs = {$s_1$,$s_2$,$s_3$...$s_n$}, vector of suspect IPs
for $s_i$ in SuspectIPs do
    Alarm.append($s_i$);
end
UpstreamSwitchList = {$d_1$,$d_2$,$d_3$...$d_m$}, vector of upstream switches
sendAlarm(UpstreamSwitchList);
```

Source: The authors (2021)

Local and Remote Contraction. So far, we have shown how pushback is adopted for the deployment of defenses. However, BUNGEE includes a reverse process to inactivate these defenses, which we refer to as *contraction*. This contraction process allows the optimal use of TCAM and SRAM resources. As shown in the FSM in Figure 4.2, we consider two contraction types. *Local contraction* refers to deactivating defenses after an attack ceases. For this, after a pushback action, the switch keeps a registry of monitoring windows without entropy alteration (*contraction_count*). When the switch considers that the attack stopped for a certain number of windows (*n*), the defenses are deactivated, and the switch returns to the initial state.

During its operation, as the switch begins to filter off a suspect IP address, it notifies the downstream neighbor that filtering has been handed off and that it can stop filtering the suspect; we denote this notification process as *remote contraction*. As this process involves another device, it occurs in different windows, similarly to what happens with the pushback process. A switch begins to filter a suspect in window W_t and notifies the downstream device. At this point, the downstream device requires receiving the same notification from *all* (differently from the pushback action) its upstreams switches to execute the contraction process, *i.e.,* remove the IP address as a suspect. Thus, this process will occur in *up to* window W_{t+w}, where w corresponds to the number of time windows the switch takes to receive contraction notifications from all its upstream switches (in the best case $w = 1$, in the worst case it is a maximum operator-defined value). Algorithm 4.4 shows how the contraction process operates in an upstream switch after receiving an alarm packet.

Figure 4.4 – Contraction process considered in BUNGEE

```
Input: Incoming packet from suspect (pkt)
       Consecutive monitoring windows without detection ($c$)

if $c == n$ then
    deactivate(defenses);
end
if pkt.SourceIP $\in$ SuspectIPs then
    ApplyFilteringStrategy(pkt);
    createNotification(pkt);
    Notification.append(pkt.SourceIP);
    DownstreamSwitchList = $\{d_1, d_2, d_3 \dots d_m\}$, vector of downstream switches
    sendNotification(DownstreamSwitchList);
end
```

Source: The authors (2021)

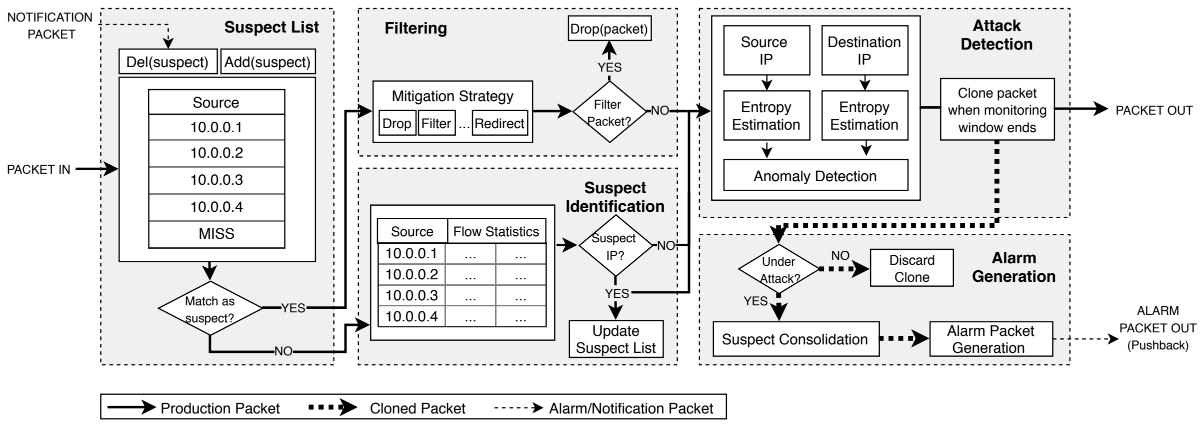
4.3 Architectural Components

The main components that comprise BUNGEE are illustrated in Figure 4.5. These are expected to be executed on the victim's switch. Alternatively, intermediate and edge switches implement slightly different functionality compared to the victim's switch. Next, we detail the role and operation of each component, following their invocation order in the pipeline followed by ingress packets. Whenever the victim's switch and the intermediate and the edge switches behave differently, those differences will be made explicit.

Suspect List. This is the first component of the mitigation mechanism. It is responsible for examining the source IP address of all incoming packets and determining if they come from sources suspect of generating attack traffic. To this end, the component implements a data structure[4] that is consulted on a per-packet basis. If the source address of an incoming packet matches any entry on the suspect list, it is selected for filtering. Otherwise, it continues

[4]This data structure is populated by the *Suspect Identification* component.

Figure 4.5 – DDoS attack detection and pushback mechanism implemented in the programmable switch.



Source: The authors (2021)

following the usual flow, to the *Suspect Identification* component.

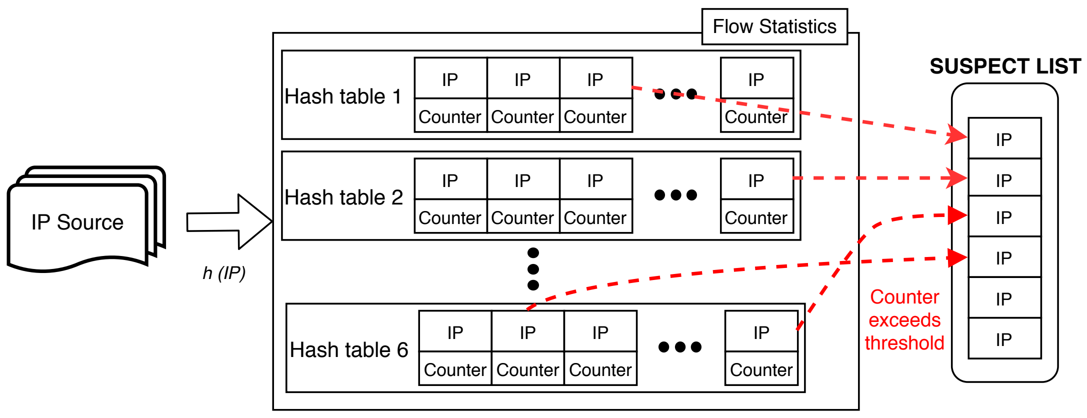
Filtering. Packets that match the suspect list must be subjected to filtering. Different strategies can be implemented, ranging from dropping all suspect packets to just a fraction of them, according to a predetermined policy (*e.g.,* random, individual, route-based, score) (Kalkan; Gür; Alagöz, 2017). In this work, we adopt a partial packet dropping procedure where we control the ratio of suspect packets that will be allowed to follow their way to the victim. Packets that are not dropped move forward in the pipeline to the *Attack Detection* component.

Suspect Identification. For the proper functioning of our mechanism, a list of suspects must be generated. As we mentioned earlier in the formalization section, we discretize traffic analysis using monitoring windows. During a window, the victim switch determines the source IP address of every incoming packet and updates counters associated with addresses. As shown in Figure 4.6, we resort to a data structure based on sketches (multiple hash tables) (SIVARAMAN *et al.*, 2017) to store the monitored statistics. At the end of a window, IP addresses whose frequency is higher than a determined threshold are considered suspects. These sources are sent to the *Suspect List* component.

In an intermediate or edge switch, this process is slightly different, as this device only keeps a list of IP addresses to inspect, whose entries are included by a downstream switch (*i.e.,* the one(s) from which it receives alarms). In these devices, source IP addresses matching the inspection list are included in the suspect list.

Attack Detection. The previous components handle the construction/maintenance of a suspect list, as well as the execution of a filtering procedure on packets from suspect IPs. For the sake of triggering the pushback mechanism, we need to detect if an intrusion is taking place.

Figure 4.6 – Sketch-based data structure for suspect identification.



Source: The authors (2021)

This is the responsibility of the *Attack Detection* component. For intrusion detection, we resort to a robust entropy-based heuristic proposed in our previous work (Lapolli; Adilson Marques; Gaspary, 2019). Additionally, the last packet within a monitoring window is *cloned* and used to trigger the consolidation of values; as such, this *cloned* packet will carry the consolidate values to the *Alarm Generation* component for analysis. A new monitoring window is initiated. All packets that reach this point leave the pipeline and are forwarded to their intended destination.

Alarm Generation. Consolidated entropy values are compared to values estimated for a "healthy" network. If the difference is higher than a determined threshold, the network is considered to be under attack and alarm packets are sent to the upstream switches. Each alarm packet carries the IP addresses considered suspects, consolidated from the *Suspect Identification* component. As previously mentioned, the implementation of this component in P4 is achieved with *cloning* and *recirculation* of packets.

For intermediate and edge switches, in addition to the alarm packets being sent upstream, the intrusion detected is also notified to the downstream switches. Further, an intermediate or edge switch that is upstream in relation to the victim and receives an alarm packet will verify if it can also detect the attack[5]. If so, packets arriving from suspect sources will be subject to filtering at this upstream location, and downstream switches will be notified to stop filtering the address (thus effectively *"pushing back"* the attack).

[5]Attack detection thresholds in intermediate or edge switches are set up to a (configurable) fraction of those used by the victim's switch. The rationale behind this is that the farther from the victim a switch is, the less aggressively an ongoing attack will be perceived (or even missed completely), since the attack traffic is potentially coming from different sources (see Figure 4.1).

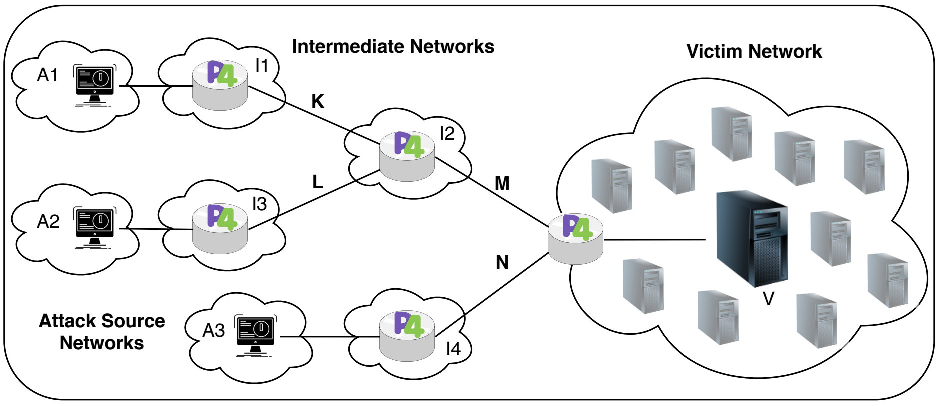
4.4 Experimental Evaluation

We performed an experimental evaluation to demonstrate the technical feasibility of BUNGEE. Since we are interested in accuracy, resource utilization, reaction time and resource footprint analysis (and not on performance measurements), we carried out an emulation-based evaluation.

4.4.1 Experiment Setup

Our test infrastructure was instantiated on a virtual machine running Ubuntu 16.04 with four processors and 16 GB RAM. Figure 4.7 illustrates the instantiated network. It is comprised of a victim network, three attack source edge networks, and four intermediate networks. The forwarding devices depicted in the figure are those that support P4 and are configured to run our mechanism using the P4 software switch (BMv2 – behavioral model version 2 (P4 LANGUAGEM CONSORTIUM, 2014)).

Figure 4.7 – Topology used in the experiments.



Source: The authors (2021)

We conducted a sensitivity analysis to determine the parameters of the proposed mechanism (artifacts available in (GONZÁLEZ *et al.*, 2022)). We report the results obtained for the best combination of parameter values, which were the following. The window size was set to 8,192 packets. The threshold to consider an IP source address as a suspect was fixed to 1% of the window size (*i.e.,* 82 packets per window). Finally, the filtering component was configured to perform traffic throttling, allowing 30% of suspect packets to follow their way to the victim.

We used the CAIDA Anonymized Internet Traces 2016 dataset (CAIDA, 2016) to represent legitimate traffic on the network. To perform the attack, we used the CAIDA DDoS Attack 2007 dataset (CAIDA, 2007), consisting of anonymized traffic traces from a DDoS attack of around one hour. It is worth mentioning that the attack detection component needs to be trained to set values for a "healthy" network, *i.e.*, calculate the entropy values for source and destination IP addresses in a "normal" operation. In the test phase (period in which our mechanism is evaluated), these values are used to determine if the network is under attack. In addition, in our evaluation, we assumed there is no IP Spoofing in the test phase. We generated training and test workloads for each of the three attack sources. The training phase was composed of 150 windows containing legitimate traffic only. As for the test phase, the workload consisted of 300 windows, where the first 75 windows were formed by legitimate traffic, the following 150 windows encompassed a mix of authentic and malicious traffic, and the 75 last windows were, again, legitimate traffic.

In our experiments, three different scenarios were considered, ranging from a confined to a broad attack, as follows:

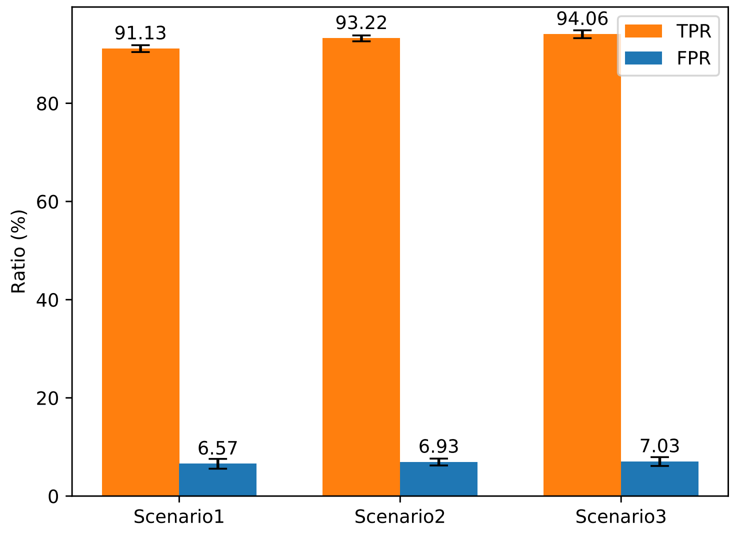
- *Scenario 1 – One attack source network:* hosts in the source network A1 send a proportion of 90% malicious-to-legitimate packets during the 150 attack windows, while hosts in the other two source networks transmit legitimate traffic all the time.
- *Scenario 2 – Two attack source networks:* hosts in source networks A1 and A2 send a proportion of 45% malicious-to-legitimate packets during the 150 attack windows, while the remaining source network sends legitimate traffic all the time.
- *Scenario 3 – Three attack source networks:* hosts in the three source networks, A1, A2, and A3, transmit each a proportion of 30% malicious-to-legitimate packets during the 150 attack windows.

We analyzed the following evaluation metrics: *detection accuracy*, *network utilization*, *reaction time*, and switch *memory footprint*. Accuracy was measured through *true positives* (ratio of packets coming from suspect IP addresses that were indeed dropped) and *false positives* (proportion of packets coming from legitimate IP addresses that were wrongly dropped). Network utilization was measured considering the amount of link bandwidth consumed by attack packets (without and with the proposed solution activated). Reaction time was calculated as the delay between the beginning of the attack and BUNGEE's filtering start. Finally, memory footprint was measured by calculating the amount of device memory that our mechanism requires to be implemented.

4.4.2 Accuracy

Figure 4.8 shows the accuracy of the proposed mechanism to detect DDoS attacks considering the three scenarios. As one can observe, the detection rate is higher than 90% for all cases. Besides, the false-positive rates are below 10% for all scenarios. The figure also reveals that the mechanism is nearly equally effective regardless of the origin (single x multiple sources) of the attack. Notice that these results are consonant with the performance of state-of-the-art approaches (*e.g.*, (Febro; Xiao; Spring, 2019; Li *et al.*, 2019)). As we will show next, the gains resulting from our proposal are not necessarily in higher accuracy (already at the top), but substantially lower usage of unnecessary network and processing resources.

Figure 4.8 – BUNGEE accuracy.

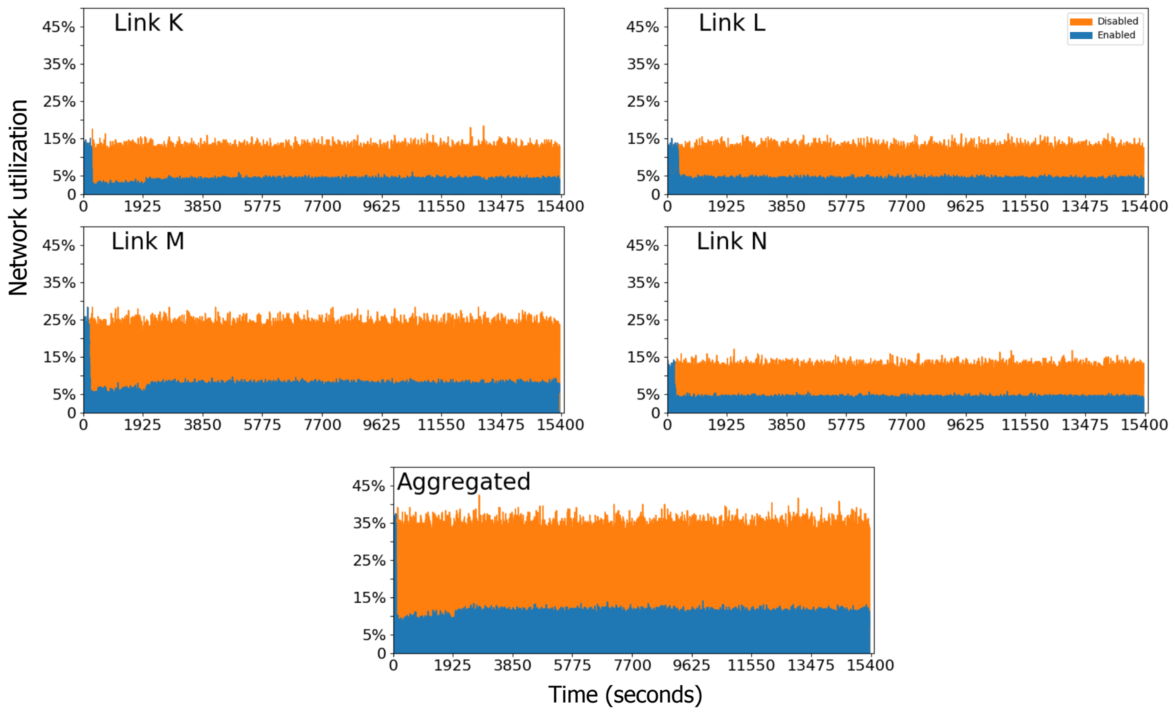


Source: The authors (2021)

4.4.3 Network Utilization

DDoS attacks cause link saturation. The attack packets end up occupying a substantial share of the links, sometimes exceeding their capacity. As a result, network resources are “wasted”, and legitimate traffic may not reach its destination. Figure 4.9 illustrates the occupation of links K, L, M, N (and aggregated) of the network topology without and with our mechanism enabled. Remember that, in this evaluation, we allow 30% of suspect packets to follow their way to the victim. As one can observe, without the mechanism, network resource consumption with attack traffic ranges from 15% to 25%. With our mechanism activated, these numbers fall to around 5%. Looking carefully at the aggregated plot, the difference between the orange and the blue areas represent the bandwidth savings obtained by our proposed mechanism, which near 65% with the given parameters.

Figure 4.9 – Network utilization savings resulting from BUNGEE.



Source: The authors (2021)

The mentioned parameters have different effects on BUNGEE: firstly, the monitoring window size affects the processing time to make decisions, changing the reaction time. Larger monitoring windows take longer to process. Secondly, the threshold used to determine if an IP is suspicious affects BUNGEE's accuracy. Finally, the probability of dropping packets in the filtering strategy will trade off overall system accuracy and network utilization.

4.4.4 Reaction Time

While packet processing delays are minimal (due to the architecture of a P4-enabled switch, designed to operate at line speed) and can be neglected, the monitoring window size plays an essential role in the time it takes for BUNGEE to activate its defensive actions. Table 4.1 shows the theoretical upper-bound for the reaction time of our mechanism as a function of window size. As one can observe, for a window of 8,192 packets, BUNGEE can detect a DDoS attack and initiate its mitigation procedures in as low as 1 millisecond (for a 1 Gbps link) and 0.1 milliseconds (10 Gbps). Even for larger window sizes, the reaction takes place in sub-second time frames. Such low values are achieved mainly due to the design of BUNGEE as a fully in-network mechanism, eliminating the periodic intervention of the control plane controller.

Table 4.1 – Reaction time to activate defensive actions.

Window Size (Packets)	1Gbps (ms)	10 Gbps (ms)
8,192	1.114	0.111
32,768	4.416	0.441
131,072	17.825	1.782
262,144	35.651	3.565
524,288	71.303	7.130

Source: The authors (2021)

4.4.5 Memory Footprint

Each forwarding device instantiates a set of data structures to handle the attack detection and the pushback mitigation mechanisms. For suspect identification, we devised a sketch-based structure (SIVARAMAN *et al.*, 2017) composed of six stages (*i.e.,* hash tables). Each hash table has 1,400 entries with 7 bytes each. This accounts for a total of 58,800 bytes. Given a monitoring window of size *ws* packets and a suspect threshold of *k* packets, *ws/k* represents the maximum number of suspects that can exist within a window. Considering our current instantiation of *ws* = 8,192 packets and *k* = 82 packets, we would need room for 100 suspects, while the data structure supports a considerably higher number of entries (8,400). As we illustrate in Table 4.2, many other configurations for both *ws* and *k* would still be supported by the hash data structure (collision-free).

In addition to the data structure described above, each switch instantiates three vectors to maintain the inspection and suspect lists, as described in Section 4.3. Each entry of these structures occupies 4 bytes to store an IP address. Considering lists of 1,400 entries each, the total number of bytes needed for them is 16,800 bytes. Finally, there is the cost of executing the entropy-based detection mechanism described in our research group previous work (Lapolli; Adilson Marques; Gaspary, 2019), which is of 39,040 bytes. Therefore, the total switch memory

Table 4.2 – Maximum number of suspects for window size and threshold.

Window Size (Packets)	Suspect Threshold (Packets)	Max. Suspects
8,192	41	200
	82	100
	123	67
32,768	41	800
	82	400
	123	267
131,072	41	3,197
	82	1,599
	123	1,066
262,144	41	6,394
	82	3,197
	123	2,132
524,288	41	Not Supported
	82	6,394
	123	4,263

Source: The authors (2021)

footprint is of 114,640 bytes, which is deemed very low, given that existing forwarding devices have available memory in the order of hundreds of MB.

5 BUNGEE-ML: A CROSS-PLANE APPROACH FOR A COLLABORATIVE DEFENSE AGAINST DDOS ATTACKS

In this chapter, we introduce BUNGEE-ML, our hybrid approach that plays on the strengths of both the data and control planes in cooperation[1] . It substantially enhances our previous work BUNGEE, described in Chapter 4, revisiting the design of some of the original components and proposing new ones. For consistency and clarity, we present BUNGEE-ML "from scratch". We start in Section 5.1 by introducing its general design. In Section 5.2, we present an overall workflow of BUNGEE-ML components. In Section 5.3, we describe lightweight techniques for detection and rapid response for the attack, which are executed in the data plane. In Section 5.4, we discuss some limitations and lessons learned from BUNGEE, which motivated us to examine and devise the hybrid design. In Section 5.5, we present the more sophisticated machine-learning-based techniques applied in the control plane, as well as the coordination scheme between the data and control planes for proper attack mitigation. In Section 5.6, we detail BUNGEE-ML's pushback technique for coordinating network devices in the mitigation task. Finally, in Section 5.7, we evaluate the proposed approach and present the obtained results.

5.1 Overview: Vertical and Horizontal Cooperation

BUNGEE-ML is a hybrid approach that combines the fast processing of the data plane and the high capacity and intelligence of the control plane for accurate attack detection and mitigation in the network. BUNGEE-ML advocates a multi-level strategy (MARNERIDES *et al.*, 2011) for ensuring the continuous operation of the network, promoting both vertical and horizontal cooperation between network elements (Figure 5.1):

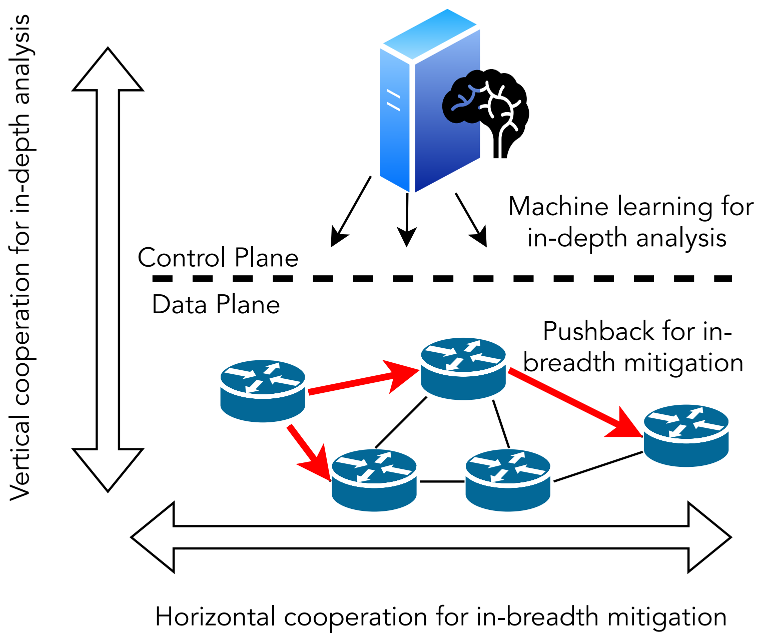
- *Vertical Cooperation*: To bypass ASIC processing limitations, BUNGEE-ML performs more sophisticated traffic analysis outside the ASIC as we (*vertically*) rely on resources from the switch CPU and from the SDN controller. This *in-depth* analysis prioritizes accuracy and can course-correct decisions made at the ASIC.
- *Horizontal Cooperation*: By leveraging the programmable topology, BUNGEE-ML opportunistically pushes back malicious traffic to as far as possible from the victim (*hor-*

[1]This chapter is based on the following manuscript under review at the *IEEE Transactions on Network and Service Management (TNSM)* journal:

- Libardo Andrey Quintero González, Lucas Castanheira, Jonatas Adilson Marques, Alberto Schaeffer-Filho, Luciano Paschoal Gaspary. **BUNGEE-ML: A Cross-plane Approach for a Collaborative Defense Against DDoS Attacks.** Manuscript ID: TNSM-2022-05131. 2022.

izontally). This is the core of our *in-breadth* mitigation strategy in the data plane and allows for quick responses to DDoS attacks.

Figure 5.1 – Vertical and Horizontal Cooperation.


Vertical cooperation for in-depth analysis
Machine learning for in-depth analysis
Control Plane
Data Plane
Pushback for in-breadth mitigation
Horizontal cooperation for in-breadth mitigation

Source: The authors (2022)

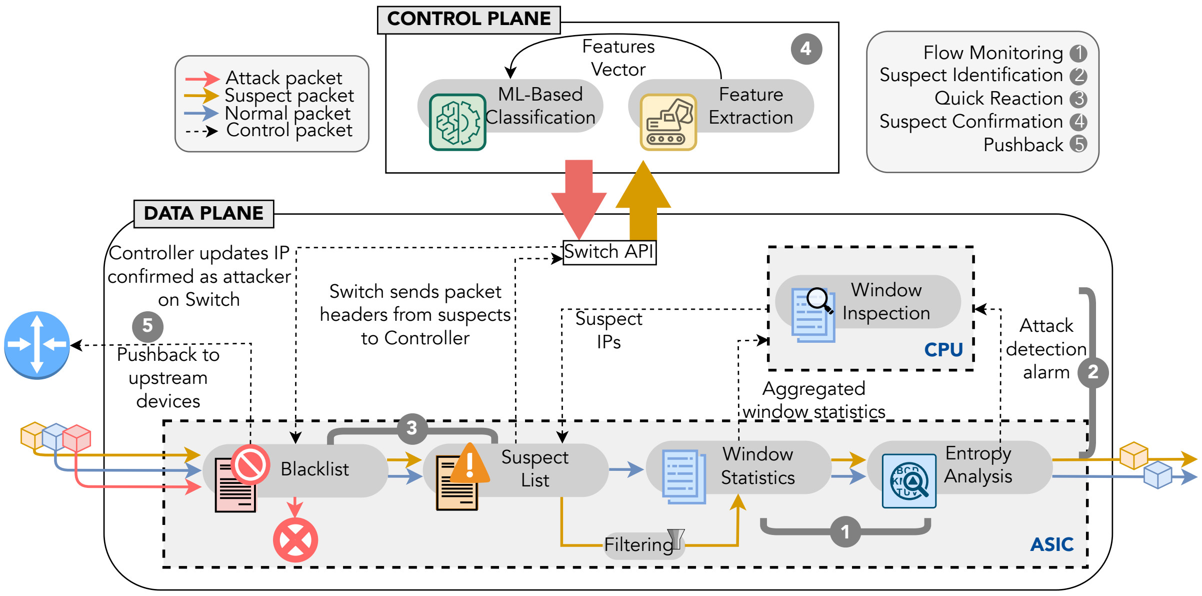
Essentially, BUNGEE-ML allows both planes to cooperate by playing on their individual strengths. First, the switch ASIC executes lightweight strategies that allow early detection of suspicious traffic at line rate. This is combined with slightly more sophisticated processing to compare recent traffic statistics at the switch CPU. However, in order to carry out a deeper, more sophisticated analysis of the suspect sources, the control plane applies machine learning techniques to decide whether suspects (identified by the data plane) are in fact attackers.

Although mitigation actions could be deployed as early as the switch has flagged a flow as suspect, e.g., the data plane can selectively throttle traffic from suspect sources to deal with the attack quickly, the countermeasures are made permanent after the control plane confirms the attacking flows. In this case, the data plane implements a pushback strategy on the confirmed suspects, prompting upstream devices to build a collaborative mitigation front and stop the attack as far as possible from the victim.

5.2 BUNGEE-ML's Workflow

Figure 5.2 illustrates the overall workflow of BUNGEE-ML, by showing the interactions between its components across the control and data planes:

Figure 5.2 – BUNGEE-ML's Workflow



Source: The authors (2022)

- The flow monitoring step (❶) is the basis of our implementation. We perform continuous monitoring and execute an strategy based on entropy analysis of incoming packets using the application-specific integrated circuit (ASIC) chips on data plane devices to detect network behavior changes during a “monitoring window”.
- In the event of an attack, the flow monitoring triggers an alert to the *Window Inspection* component (❷). In this component, the most recent monitoring window source addresses are compared with global statistics to identify the suspects of causing the disturbance in the network. For this purpose, we resort to a general-purpose processor (e.g., x86 CPU) attached to the switch to manage these statistics and store them out of the fast path.
- After suspect sources are identified, a quick reaction begins (❸). This includes maintaining a *Suspect List* so that subsequent packets from suspects are filtered. Incoming suspect packets are diverted to the control plane, where further inspection is carried out to determine whether sources are attackers or not.
- The control plane extracts packet features to classify source addresses using machine

learning mechanisms (❹). After an address is classified, the control plane notifies the data plane to *(a)* remove the addresses classified as benign from the *Suspect List* or *(b)* confirm the addresses classified as malicious, *i.e.,* include them in a *Blacklist*. Incoming packets from sources included in the *Blacklist* are dropped.

- Finally, the switch alerts its first-hop upstream devices about the ongoing attack (❺) by sending the list of confirmed suspects to take the appropriate mitigation actions, which we refer to as a pushback action.

We highlight that steps ❶, ❷, ❸, and ❺ are all executed in the data plane for detecting and mitigating an attack. In the meantime, the control plane refines the ongoing list of suspects formed by the data plane (❹) to improve the classification and mitigation accuracy. Next, we describe the operation of BUNGEE-ML in more details.

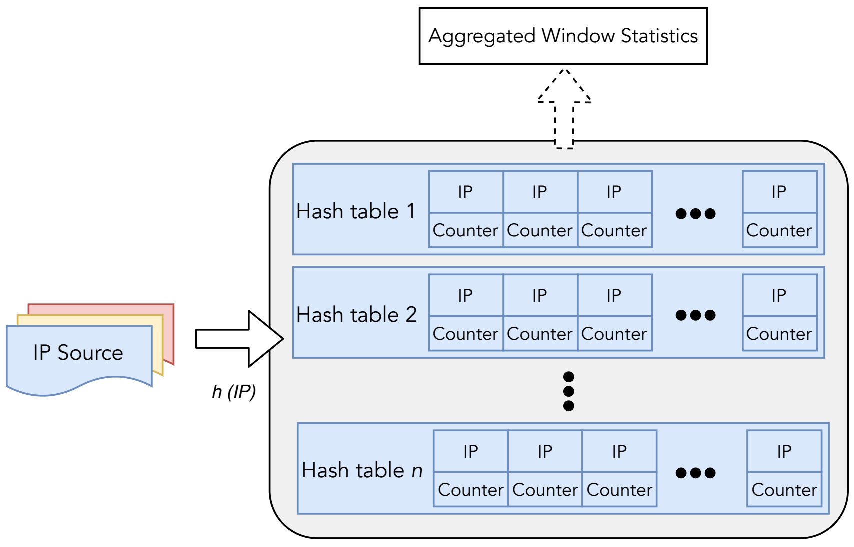
5.3 Data Plane Implementation

Most of the components of BUNGEE-ML are carried out by the data plane. The implementation of these components alone already enables BUNGEE-ML to react to DDoS attacks to some extent. In this section, we describe the components that are executed in the data plane, except for the pushback procedure, which will be presented in Section 5.6.

5.3.1 Flow Monitoring

To be able to detect attacks and classify malicious sources, we first need to analyze the network traffic. We discretize traffic analysis using monitoring windows. We rely on an *Entropy Analysis* component, which resorts to a robust entropy-based heuristic described in detail in Section 4.2 (as part of BUNGEE). At the end of the monitoring window, the entropy of the frequency of all source IP addresses is calculated, and an attack detection is assumed when this value is outside of certain expected bounds defined based on the traffic history. During a monitoring window, the switch closer to the victim determines the source IP address of every incoming packet and updates counters associated with the addresses. As shown in Figure 5.3, we resort to a data structure based on sketches (SIVARAMAN *et al.*, 2017) to store the monitored statistics in the *Window Statistics* component. At the end of a monitoring window, the statistics are aggregated, and consolidated values, as well as the result of the entropy analysis, are sent to the *Window Inspection* component for analysis.

Figure 5.3 – Sketch-based data structure statistics aggregation.



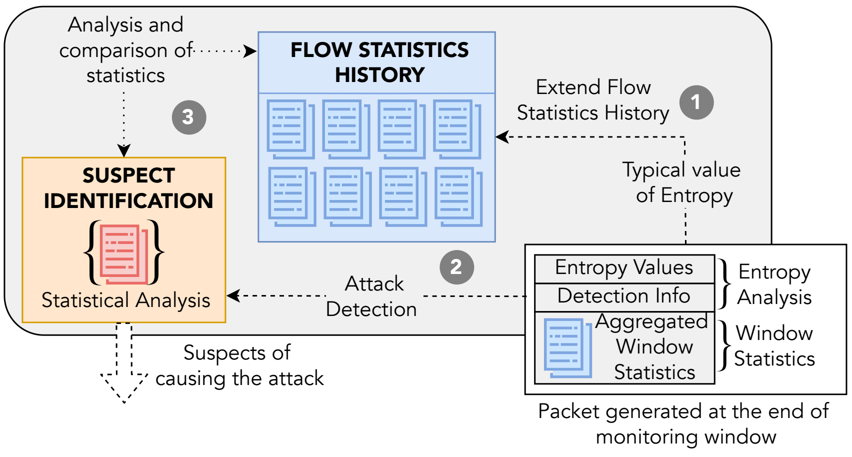
Source: The authors (2022)

5.3.2 Statistics-Based Suspect Identification

BUNGEE-ML identifies suspects by analyzing their behavior during a long period and recognizes sources causing the entropy disturbance. As shown in Figure 5.4, when the *Entropy Analysis* shows a typical result (❶), the recent window statistics are stored to serve as a comparison with future windows. Due to storage and processing constraints, it is not easy to make this using the ASIC chips on devices. As such, we resort to placing this component at the general-purpose processor attached to the switch (*e.g.*, x86 CPU), which gives more flexibility for what we require. This CPU can quickly access and interact with the ASIC chip without impacting the demanded processing speed.

In the opposite case – i.e., if the *Entropy Analysis* reports a significant change (*i.e.,* attack detection (❷)) – the monitoring window is submitted to a process of analysis and comparison with the previously stored statistics for suspect identification (❸). Statistics of the monitoring window are compared with values adjusted according to the environment (calculated as the previous windows statistics are stored). Aligned with the literature, we use statistical heuristics to calculate these values dynamically (HONG; LEE; LEE, 2019). Flow statistics history are used to calculate the mean (μ) and the standard deviation (δ) of the number of packets sent per source in a window. When a window is received with attack detection, these values are used in Equation 5.1 to define the maximum acceptable number of packets sent per source in a window.

Figure 5.4 – Statistics aggregation at the end of a monitoring window.


Analysis and comparison of statistics
FLOW STATISTICS HISTORY
Extend Flow Statistics History
1
3
Typical value of Entropy
SUSPECT IDENTIFICATION
Statistical Analysis
2
Attack Detection
Entropy Values
Detection Info
Aggregated Window Statistics
Entropy Analysis
Window Statistics
Suspects of causing the attack
Packet generated at the end of monitoring window

Source: The authors (2022)

$$freq_{max} = \mu + (k \cdot \sigma) \tag{5.1}$$

In Equation 5.1, *k* is a configurable parameter that denotes the suspect identification thresholds – a lower value of *k* represents more rigorous identification conditions, while a high value represents a more relaxed rule. A source is considered as a suspect whenever any of these two conditions hold:

- The source has not been seen in any of the previous windows stored.
- The statistics show that a source is sending more packets than the calculated $freq_{max}$ value.

After executing the statistical-based analysis, the next step is to include all of the suspect sources identified in the *Suspect List* component, described next.

5.3.3 DDoS Rapid Response

We proceed to deal with the packets incoming from suspects to defend the victim. We consider two mechanisms that comprise the reaction to the attack. The first is the *Suspect List*, responsible for examining the source IP address of all incoming packets and determining if they come from a suspect source. It implements a data structure consulted on a per-packet basis. If the source address of an incoming packet matches any entry on the *Suspect List*, it is selected for filtering. Otherwise, it continues following the usual flow to the *Window Statistics* component. Given the trade-offs among the techniques presented in Section 2.4, our proposed solution employs *packet filtering* for a DDoS rapid response. Different strategies can be implemented for the filtering strategy, ranging from dropping all suspect packets to just a fraction of them, according to a predetermined policy (*e.g.,* random, individual, route-based, score) (Kalkan; Gür; Alagöz, 2017). Packet filtering can be adequately parameterized to be more conservative or relaxed, depending on the protected services (and underlying infrastructure). In particular, we adopt a partial packet dropping procedure where we control the ratio of suspect packets allowed to follow their way to the victim. Packets that are not dropped move forward in the pipeline.

The *Suspect List* works similarly to the one presented previously in Section 4.3, with the difference that in BUNGEE-ML, it interacts with the (switch) CPU and the control plane controller (for accurate detection of malicious sources, see Figure 5.2). Regardless of whether the incoming packets from suspect sources are filtered or not, upon arrival, they are cloned, and the header is used to carry out an in-depth analysis (which will be described in Section 5.5). The result of this analysis serves to create the second mechanism, which we refer to the *Blacklist*. It is similar to the *Suspect List* but, in this case, incoming packets from specific sources are dropped. Furthermore, only sources from the *Blacklist* are considered for the pushback process (Section 5.6).

5.4 Overcoming Data Plane Limitations

As we described before, our previous work BUNGEE (Chapter 4) runs entirely on the data plane, using several P4 primitives (i.e., `clone` and `recirculate`) (CONSORTIUM, 2020) to detect and mitigate a DDoS attack. Our experience developing a data plane-only DDoS mechanism led us to find data plane resource restrictions to be prohibitive in terms of space for the several data structures required. In addition, we observed a processing constraint (we need to go through all the indexes of the data structures for data analysis, increasing the volume and processing time). All of these together prevent the implementation of the detailed analysis of the network conditions, which subsequently impact mitigation decision-making.

Based on the observed limitations, in BUNGEE-ML, we propose improvements involving the switch CPU and the control plane to achieve more accurate suspect identification. One important design decision was to maintain data plane friendly operations requiring processing at line rate for DDoS rapid response and execute more complex operations in the control plane asynchronously. Thus, the non-ASIC part of BUNGEE-ML accounts for the following: we employ the switch CPU to keep the statistics history for all the monitoring windows, which alleviates the switch memory bottleneck. Moreover, we execute processes to analyze the behavior of a monitoring window in relation to several previous windows and thus discover the IP sources responsible for generating the changes in network behavior. Finally, as the control plane can run more sophisticated analysis, we invoke it to confirm whether a suspect is an attacker or not. So, we run machine learning models in the control plane, which improves detection accuracy as a background process without added latency.

5.5 Hybrid Control and Data Plane Operation

We have shown how DDoS attack detection is performed in the data plane, and a quick reaction is taken on candidate suspects of causing the attack. We incorporate the control plane in BUNGEE-ML to execute the machine learning process and improve this mitigation process. Once the switch identifies a set of suspects, packets incoming from them are submitted to an extensive analysis to confirm or release the source as a suspect.

5.5.1 Flow Classification

The control plane hardware provides the optimal place for executing in-depth analyses, such as those needed for a more accurate classification of suspects. After a first iteration to identify suspects based on *aggregated statistical data*, we now proceed to evaluate those suspects according to each *isolated* packet they send into the network. We implement a feature selection technique to rank and extract the most relevant features from the dataset comprised of a suspect's collected packet headers to produce a features vector that serves as input for the machine learning (ML) technique. With the features collected from the packet, the control plane executes the ML technique to classify the source address as a benign or malign source. For the sake of flow classification, in our approach, we employ two different ML techniques to contrast the results obtained, Random Forest and Naive Bayes. We found that Random Forest is a widely known and used technique that, employing decision trees, determines whether or not a source is malign. We use this technique for, among others, its low cost of implementation and efficiency in the class allocation. We also use Naive Bayes, a technique with a different approach based on the independence of variables to classify the analyzed sources. We do not restrict the ML techniques used (*e.g.*, those presented in Section 2.3), being possible to employ other ML techniques for source classification.

5.5.2 Information Exchange with the Data Plane

For the sake of triggering ML analysis, the control plane needs to receive the packets from the suspects. For this reason, the data plane sends a copy of the protocol headers of packets incoming from sources matching the *Suspect list*. The data plane incorporates a packet cloning policy to control the number of cloned packets sent to the controller. With this, it is possible to regulate the processing that the cloning incorporates in the data plane and prevent overloading the communication channel with the control plane. The policy can be set to i) clone all packets or ii) clone a sampling of the packets from suspect sources. Cloned packet headers are passed to the *Feature Extraction* component, where models such as Chi-Squared, Information Gain, and others are used to extract the main features from the cloned packet (*e.g.*, source and destination address, protocol, length). A vector is created with the set of features extracted by the model, which the ML technique analyzes to classify the source. Finally, when the analysis is concluded, the control plane removes the address from the suspect list on the switch when a source is classified as benign. In contrast, when the source is classified as malicious, the control plane includes the source in the Blacklist, indicating that all packets from that source should be dropped.

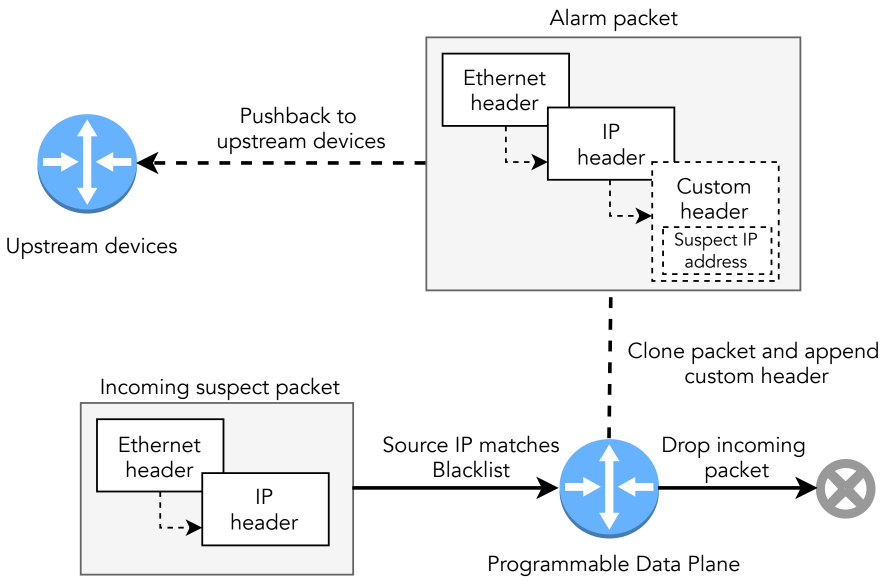
5.6 Pushback Technique

Following the idea introduced in BUNGEE to pushback defenses progressively at upstream devices, BUNGEE-ML executes the same procedure to push attackers as far away as possible from the victim. After suspect confirmation, the switch closer to the victim alerts its first-hop upstream counterparts, sending them an *alarm packet* containing the confirmed IP addresses. Upon receiving an alarm, upstream neighbors will begin dropping packets incoming from the notified IP addresses.

We execute the pushback process in the data plane by way of the P4 `clone` primitive, which allows making a copy of a packet. Thus, pushback is executed by cloning the packet which source IP matches the blacklist and using it for inter-switch communication. The cloned packet is transformed into an *alarm packet* by appending a custom header and the confirmed suspect IP. Finally, the alarm packet is sent to upstream devices (Fig. 5.5).

In addition to the described pushback process, BUNGEE-ML includes a reverse process to deactivate the defenses, which we refer to as *contraction*. This contraction process allows a more efficient use of TCAM (Ternary Content-Addressable Memory) and SRAM (Static Ran-

Figure 5.5 – Packet cloning and transforming into *alarm packet*.



The authors (2022)

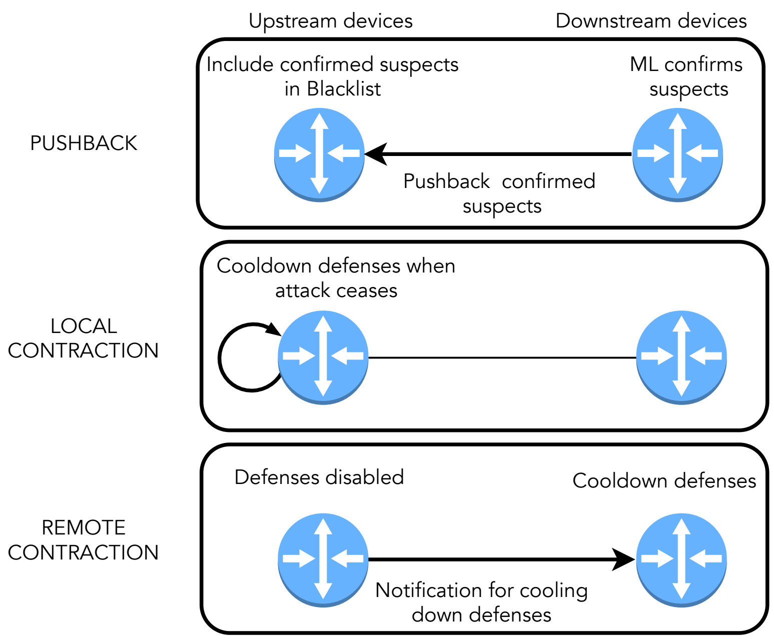
dom Access Memory) resources. As shown in Fig. 5.6, we consider two contraction types. *Local contraction* refers to deactivating defenses after an attack ceases. When the switch considers that the attack has stopped (a certain number of windows without anomalous entropy), the defenses are deactivated, and the switch returns to the initial state (*i.e.,* clear the suspect and the blacklist).

During its operation, as upstream switches drop IP addresses, they notify the downstream neighbor that mitigation has been handed off and that it can stop handling the confirmed suspect; we denote this notification process as *remote contraction*. However, the downstream device requires receiving the same notification from *all* of its upstream switches (differently from the pushback action) to execute the deactivation process, *i.e.,* remove the IP address from the blacklist.

5.7 Experimental Evaluation

We conducted an experimental evaluation to demonstrate the technical feasibility of BUNGEE-ML (artifacts available in (GONZÁLEZ *et al.*, 2022)). We carried out an emulation-based evaluation to demonstrate the gains of our approach based on classification metrics (accuracy and precision) and computational resource utilization.

Figure 5.6 – Pushback process and contraction types.


Upstream devices
Downstream devices
PUSHBACK
Include confirmed suspects in Blacklist
ML confirms suspects
Pushback confirmed suspects
LOCAL CONTRACTION
Cooldown defenses when attack ceases
REMOTE CONTRACTION
Defenses disabled
Cooldown defenses
Notification for cooling down defenses

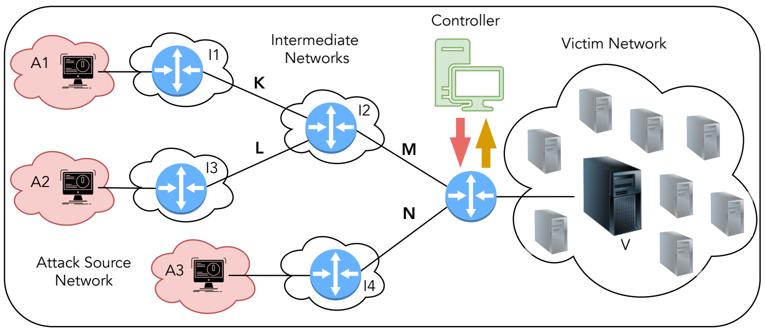
The authors (2022)

5.7.1 Experimental Setup and Dataset

Our test infrastructure was instantiated on a virtual machine running Ubuntu 16.04 with four processors and 16 GB RAM (similar to the one presented in Section 4.4.1 to evaluate BUNGEE). Figure 5.7 illustrates the instantiated topology. It comprises a victim network, four intermediate networks, and three attack source edge networks. The forwarding devices depicted in the figure are those that support P4 and are configured to run our mechanism using the BMv2 software switch. The infrastructure is also composed of an external SDN controller employed for in-depth analysis of suspect traffic using machine learning techniques.

In our experiments, we fixed the monitoring window size in 8,192 packets. For our statistical-based suspect identification, we set the value of k to 3, so $freq_{max}$ (Equation 5.1) is given by $\mu + 3\sigma$. We chose this threshold to identify the highest amount of malicious packets (here, we did not focus on penalized legitimate packets, as a refined suspect identification process will be carried out with the help of the control plane). The filtering component was configured to perform traffic throttling, allowing 30% of suspect packets to follow their way to the victim. Our evaluation combines classification techniques (*i.e.,* Naive Bayes and Random Forest) with different feature selection algorithms (*i.e.,* Chi-Square, Information Gain, and Gain Ratio) since we are interested in determining the best combination of techniques to improve

Figure 5.7 – Topology used in the experiments.



The authors (2022)

suspect identification.

We reproduced the attack scenario using the CICDDoS2019 dataset (SHARAFALDIN *et al.*, 2019). In particular, this dataset contains a high proportion of attack packets (*i.e.,* the attack is quite aggressive), which allows our mitigation approach to be highly stressed. The dataset comprises two days of network traffic. We highlight that the attack detection component and ML models need to be trained to set values for a "healthy" network, *e.g.*, calculate the entropy values for source and destination IP addresses in a "normal" operation. In the validation phase (period in which our mechanism is evaluated), these values are used to determine if the network is under attack. We used the first-day network traffic to train the *Entropy Analysis* component and the ML *Flow Classification* techniques. The second day was used as the validation set of our approach. Our feature extraction algorithms selected the most representative features from the CICDDoS2019 dataset. The features adopted in our experiments are listed in Table 5.1. In addition, in our evaluation, we assumed there is no IP Spoofing in the validation phase.

5.7.2 Data and Control Plane Cooperation Gains

5.7.2.1 Accuracy and Precision of Suspects Identification

We evaluated our approach by calculating classification metrics to determine the effectiveness and robustness of BUNGEE-ML suspect identification (comprised of the lightweight mechanism running in the data plane and the execution of intelligent techniques in the controller) since this is a fundamental component of our attack mitigation mechanism. The classification metrics were calculated based on the confusion matrix for each combination of feature selector and classifier used on the ML component. We also calculated metrics for the statistical-based

Table 5.1 – Selected features from the CICDDoS2019 dataset.

No.	Feature	Description
1	src_bytes	Number of data bytes transferred from source to destination
2	service	Service used by destination network
3	dst_bytes	Number of data bytes transferred from destination to source
4	flag	Status of the connection (Error or Normal)
5	diff_srv_rate	Percentage of connections that were to the different services
6	same_srv_rate	Percentage of connections that were to the same services
7	dst_host_srv_count	Number of connections having same port number
8	dst_host_same_srv_rate	Percentage of connections that were to the same service among the connections aggregated in dst_host_count
9	dst_host_diff_srv_rate	Percentage of connections that were to different service among the connections aggregated in dst_host_count
10	logged_in	Shows login status (1- successful login, 0- otherwise)
11	dst_host_serror_rate	Percentage of connections that have activated flag (4) s0,s1,s2 or s3, among the connections aggregated in dst_host_count
12	srv_serror_rate	Percentage of connections that have activated flag (4) REJ, among the connections aggregated in srv_count
13	serror_rate	Percentage of connections that have activated flag (4) s0,s1,s2 or s3, among the connections aggregated in count (23)
14	srv_count	Number of connection to the same service (port number)
15	protocol_type	Protocol used
16	dst_host_count	Number of connections having the same destination host IP Address
17	duration	Length of the time duration of the connection
18	count	Number of connections to the same destination host
19	land	If source and destination port no. and IP addresses are same then it will set as 1 otherwise 0
20	urgent	Number of urgent packets (these packets with urgent bit activated)

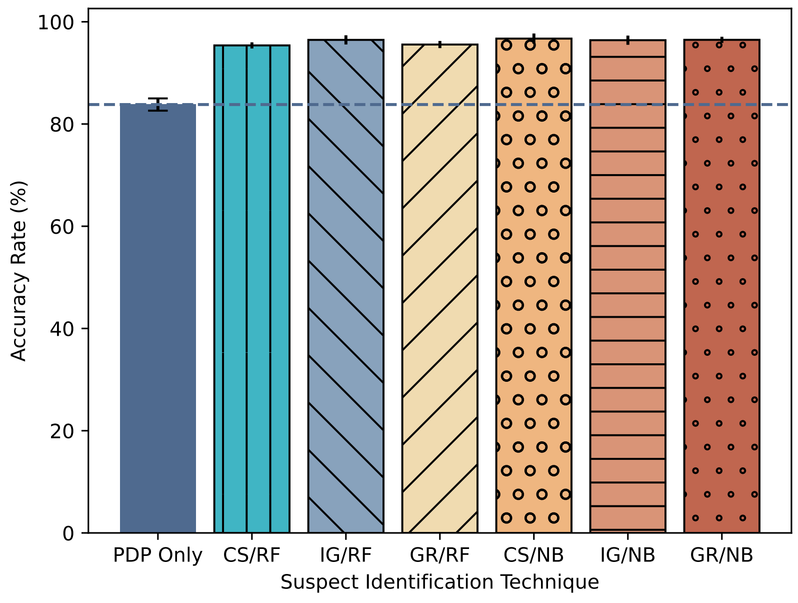
Source: The authors (2022)

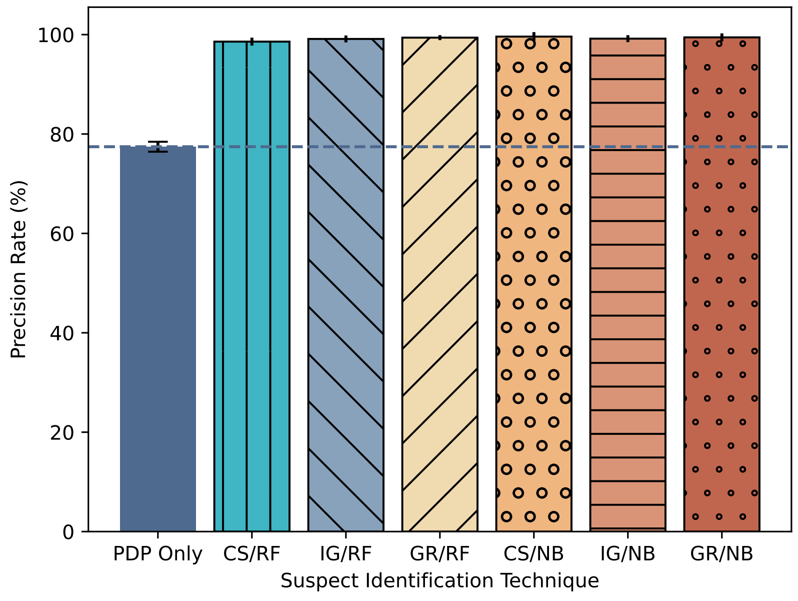
suspect identification process executed in the data plane since it is the input for machine learning and serves as the baseline result.

Figure 5.8 shows the results we obtained for BUNGEE-ML. Accuracy is the ratio between the sources (benign and malign) correctly classified and the total number of all sources. Considering only the statistical-based suspect identification process executed in the data plane, accuracy is slightly higher than 80%. This result is mainly due to the misclassification of benign sources as malign (false positives). Fortunately, these results are substantially improved with the addition of the control plane ML-based in-depth traffic analysis, in which BUNGEE-ML accuracy is above 95% for all scenarios. Precision is the ratio between correctly malign sources classified and the total number of all suspect sources classified as malign. In terms of precision, we obtained results similar to those observed for accuracy. Due to established conditions (mainly the one that conservatively considers new sources as suspects), statistical-based suspect identification recognizes a considerable amount of benign sources as suspects. This negatively affects precision, which despite not being more than 80%, it is a perfectly valid result as a first, quick mitigation procedure. After the data plane submits suspect sources to ML techniques, the control plane releases benign sources from filtering processes, reflecting in BUNGEE-ML precision higher than 99%.

Figure 5.8 – Classification metrics for the Machine Learning algorithms adopted in BUNGEE-ML flow classification.

Besides the PDP statistical-based identification, each bar corresponds to a specific combination of techniques for feature selection and classification algorithm. CS: Chi-Square, IG: Information Gain, GR: Gain Ratio, RF: Random Forest, and NB: Naive Bayes.





Source: The authors (2022)

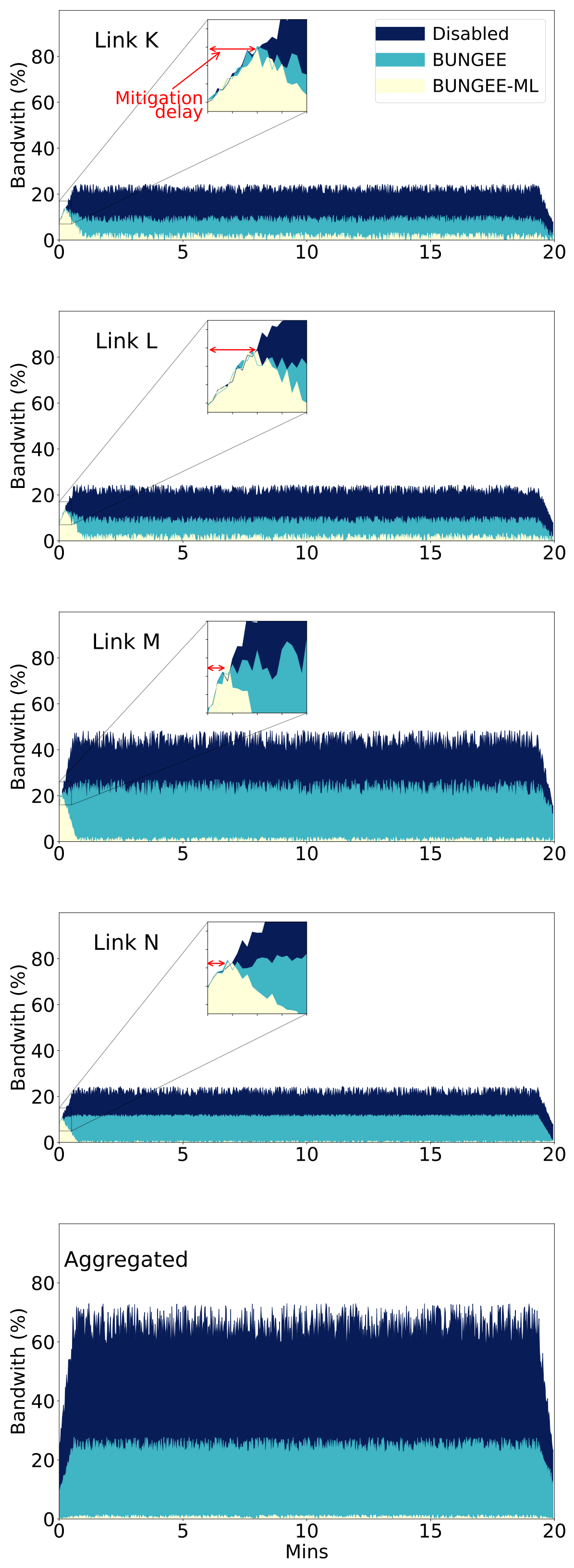
As expected, the experiments showed that the invocation of the control plane provided better results for both evaluated classification metrics, compared to the use of a data plane-based technique alone. The main advantage of BUNGEE-ML incorporating a technique executed in the data plane is that it offers a quick response while not depending on the control plane to initiate mitigation actions. This ensures data plane independence, while background control plane invocation helps in suspect confirmation. The execution of this process in the control plane offers high accuracy and precision, which triggers the pushback strategy for resource savings in communication channels, as described next.

5.7.2.2 Network Utilization

Our DDoS rapid response mitigates the attack on the switch closer to the victim, but our idea goes further and attempts to move the attack away as much as possible from the victim. Figure 5.9 shows the occupation of links K, L, M, N, and Aggregated (the link that connects the victim with the nearest switch) of the network topology without and with our mechanism enabled. Our approach pushes back only confirmed attack traffic (*i.e.,* after suspects are confirmed by the control plane). For the sake of comparison, we also show the bandwidth savings obtained when only the data plane-based strategy is activated (*i.e.,* our previous work BUNGEE (Chapter 4)). On attack detection (time 0), the switch closer to the victim already begins mitigating the attack by permanently monitoring and analyzing suspects. Remember that during the quick reaction we allow 30% of suspect packets to follow their way to the victim. As one can observe, for the aggregated link, the data plane-based strategy reduces the link usage from 70-75% to 23-27%. Furthermore, when the control plane, ML-based component is also employed, packets arriving from confirmed suspects are entirely dropped, reducing the bandwidth used by the attack to a minimal fraction.

The evaluation revealed an (expected) delay for the upstream switches to start the mitigation and, ultimately, our mitigation approach to be effective in those devices. The delay corresponds to the time it takes for switches to be notified of the attack and receive the IP addresses to initiate the mitigation. This time is proportional to how far devices are from the victim switch, as the pushback process is executed gradually. We show these delays with a zoom square on the K, L, M, and N links in Figure 5.9. It is also worth observing that the usage of the corresponding downstream link decreases suddenly as soon as a switch is notified and starts mitigation.

Figure 5.9 – BUNGEE-ML network utilization savings resulting from the proposed mechanism.



The authors (2022)

5.7.3 Data and Control Plane Overhead

5.7.3.1 Data Plane Memory Footprint

Our approach requires each device to instantiate a set of data structures to handle the detection and attack mitigation mechanisms. The memory footprint for executing the entropy-based attack detection mechanism is 39,040 bytes, as described in a previous work by our research group (Lapolli; Adilson Marques; Gaspary, 2019). The sketch-based structure adopted for suspect identification comprises six stages (*i.e.,* hash tables). Each hash table is configured to store 1,400 entries with 7 bytes each. In this way, it is required 58,800 bytes to operate the sketch-based structure. The attack reaction includes a Suspect List and Blacklist that maintain IP addresses marked and confirmed as suspects. The cost of maintaining these lists is 11,200 bytes. Therefore, the total switch memory footprint is 109,040 bytes, which is deemed very low, given that existing forwarding devices have available memory in the order of hundreds of MB.

5.7.3.2 Machine Learning Overhead

As evidenced in Section 5.7.2.1, BUNGEE-ML offers high accuracy and precision when executing attack mitigation. We have measured the computational resource consumption of the ML techniques used in our experiments from when the attack is detected until it ends. The experiments included the amount of memory used and the wall clock time for the process of ML analysis. Table 5.2 shows the peak memory usage for each combination of Feature Selection and ML technique.

Table 5.2 – Peak memory for each combination of Feature Selection and ML technique.

Feature Selection/ML combination	Random Forest	Naive Bayes
Chi-Squared	1,695 MB	1,783 MB
Information Gain	1,428 MB	1,502 MB
Gain Ratio	1,574 MB	1,586 MB

Source: The authors (2022)

In the worst scenario, the peak memory usage was 1,783 MB for the Chi-Squared/Naive Bayes combination, while the Information Gain/Random Forest combination resulted in the lowest memory use of 1,428 MB. Wall clock time is the time elapsed between a suspect notification sent to the control plane controller for analysis and the classification output as a benign or malign source. In Table 5.3, we list wall clock time results obtained for each combination of Feature Selection and ML technique.

Table 5.3 – Wall clock time for each combination of Feature Selection and ML technique.

Feature Selection/ML combination	Random Forest	Naive Bayes
Chi-Squared	304 ms	315 ms
Information Gain	309 ms	319 ms
Gain Ratio	299 ms	322 ms

Source: The authors (2022)

The maximum wall clock time observed was 322 ms using the Gain Ratio/Naive Bayes combination. The lowest wall clock time of 299 ms was observed for the Gain Ratio/Random Forest combination. These numbers evidence the low resource footprint demanded by BUNGEE-ML.

6 CONCLUSION

In this work, we explored the Software-Defined Networks concept to devise novel solutions to detect and mitigate DDoS attacks. Our work unfolded into two main contributions: BUNGEE and BUNGEE-ML. BUNGEE is a mechanism for DDoS detection and mitigation to be entirely executed on a programmable P4 data plane. Our main research contribution was the proposal of an adaptive pushback approach that, through entropy-based anomaly detection and inter-switch communication, allows for the optimal, dynamic deployment of filtering procedures. The results obtained reveal the mechanism has satisfactory accuracy and contributes to savings of network resources. Moreover, the solution reacts in a fraction of the time typically demanded by external CPU and SDN controller-based counterparts. Finally, it consumes a small amount of a forwarding device's memory, enabling other applications to co-exist on the device.

As an evolution to our seminal work on BUNGEE, we proposed BUNGEE-ML, a hybrid approach for DDoS detection and mitigation with the interaction of a programmable P4 data plane and a control plane controller. BUNGEE-ML employs a lightweight mechanism for rapid attack detection and mitigation in the data plane. In the background, BUNGEE-ML employs vertical cooperation with the control plane controller for in-depth analysis of suspects (executing suitable machine learning techniques for traffic classification). This way, our approach refines mitigation actions executed in the data plane, improving the accuracy of mitigation actions against suspects. Additionally, by adopting a pushback strategy, our work promotes collaborative mitigation with the horizontal coordination between switches, allowing in-breadth mitigation. Our main novel research contribution was the proposal of a hybrid approach combining the fast processing of the data plane and the high capacity and intelligence of the control plane for attack detection and mitigation. Our evaluation results demonstrate that BUNGEE-ML is highly accurate in suspect identification (higher than 99%), giving us a degree of confidence to apply packet dropping on confirmed suspects. In addition, our pushback mitigation strategy helps avoid the waste of network bandwidth. Finally, it is worth mentioning that BUNGEE-ML incurs a low resource footprint, which allows it to be considered for use in practice.

Given the relevance of the area and the sophistication of data plane-based security solutions, as is the case of BUNGEE-ML, there are research avenues worth exploring as future work. One of our objectives is to experiment BUNGEE-ML considering different physical targets and varying DDoS scenarios. Moreover, we expect to advance our ML-based approach toward providing the operator with explainability functionality.

REFERENCES

Agrawal, N.; Tapaswi, S. Defense mechanisms against ddos attacks in a cloud computing environment: State-of-the-art and research challenges. **IEEE Communications Surveys Tutorials**, p. 1–1, 2019.

ALSADI, A. *et al.* A security monitoring architecture based on data plane programmability. In: **2021 Joint European Conference on Networks and Communications 6G Summit (EuCNC/6G Summit)**. [S.l.: s.n.], 2021. p. 389–394.

ARSHI, M.; NASREEN, M.; MADHAVI, K. A survey of ddos attacks using machine learning techniques. **E3S Web of Conferences**, v. 184, p. 01052, 01 2020.

ATLAS, N. A. **14th Annual Worldwide Infrastructure Security Report (WISR)**. 2019. Disponível em: <https://www.netscout.com/report/>.

BAWANY, N. Z.; SHAMSI, J. A.; SALAH, K. Ddos attack detection and mitigation using sdn: Methods, practices, and solutions. **Arabian Journal for Science and Engineering**, v. 42, n. 2, p. 425–441, Feb 2017. ISSN 2191-4281. Disponível em: <https://doi.org/10.1007/s13369-017-2414-5>.

BHARDWAJ, A. *et al.* Ddos attacks, new ddos taxonomy and mitigation solutions—a survey. In: IEEE. **2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES)**. [S.l.], 2016. p. 793–798.

BOSSHART, P. *et al.* P4: Programming protocol-independent packet processors. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 44, n. 3, p. 87–95, jul. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2656877.2656890>.

Bülbül, N. S.; Fischer, M. Sdn/nfv-based ddos mitigation via pushback. In: **ICC 2020 - 2020 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2020. p. 1–6. ISSN 1938-1883.

CAIDA. **The CAIDA UCSD "DDoS Attack 2007" Dataset**. 2007. Disponível em: <https://www.caida.org/data/passive/ddos-20070804_dataset.xml>.

CAIDA. **Anonymized Internet Traces 2016**. 2016. Disponível em: <https://catalog.caida.org/details/dataset/passive_2016_pcap>.

CARVALHO, R. *et al.* Enhancing an sdn architecture with dos attack detection mechanisms. **Advances in Science, Technology and Engineering Systems Journal**, v. 5, p. 215–224, 01 2020.

CARVALHO, R. N. *et al.* Detecting ddos attacks on sdn data plane with machine learning. In: **2021 Ninth International Symposium on Computing and Networking Workshops (CANDARW)**. [S.l.: s.n.], 2021. p. 138–144.

CONSORTIUM, T. P. L. **P4_16 Language Specification**. 2020. Disponível em: <https://p4.org/p4-spec/docs/P4-16-v1.2.1.pdf>.

CORDEÍRO, W. L. C.; MARQUES, J. A.; GASPARY, L. P. Data plane programmability beyond openflow: Opportunities and challenges for network and service operations and management. **J. Netw. Syst. Manage.**, Plenum Press, USA, v. 25, n. 4, p. 784–818, oct 2017. ISSN 1064-7570. Disponível em: <https://doi.org/10.1007/s10922-017-9423-2>.

DALAL, K. R. Analysing the role of supervised and unsupervised machine learning in iot. In: **2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)**. [S.l.: s.n.], 2020. p. 75–79.

DALMAZO, B. L. *et al.* A systematic review on distributed denial of service attack defense mechanisms in programmable networks. **Int. J. Netw. Manag.**, John Wiley amp; Sons, Inc., USA, v. 31, n. 6, nov 2021. ISSN 1099-1190. Disponível em: <https://doi.org/10.1002/nem.2163>.

Dargahi, T. *et al.* A survey on the security of stateful sdn data planes. **IEEE Communications Surveys Tutorials**, v. 19, n. 3, p. 1701–1725, 2017.

DAWSON, C. **Amazon invited DDOS attack on Prime Day**. 2018. [Online; accessed 24-September-2021]. Disponível em: <https://tamebay.com/2018/07/amazon-invited-ddos-attack-on-prime-day.html>.

DIMOLIANIS, M.; PAVLIDIS, A.; MAGLARIS, V. Signature-based traffic classification and mitigation for ddos attacks using programmable network data planes. **IEEE Access**, p. 1–1, 2021.

DING, D.; SAVI, M.; SIRACUSA, D. Tracking normalized network traffic entropy to detect ddos attacks in p4. **IEEE Transactions on Dependable and Secure Computing**, p. 1–1, 2021.

DONG, S.; ABBAS, K.; JAIN, R. A survey on distributed denial of service (ddos) attacks in sdn and cloud computing environments. **IEEE Access**, v. 7, p. 80813–80828, 2019.

DUNGAY, D. **Software Defined Networking Explained**. 2019. Disponível em: <https://www.commsbusiness.co.uk/features/software-defined-networking-sdn-explained/>.

FEAMSTER, N.; REXFORD, J.; ZEGURA, E. The road to sdn: An intellectual history of programmable networks. **SIGCOMM Comput. Commun. Rev.**, ACM, New York, NY, USA, v. 44, n. 2, p. 87–98, abr. 2014. ISSN 0146-4833. Disponível em: <http://doi.acm.org/10.1145/2602204.2602219>.

Febro, A.; Xiao, H.; Spring, J. Distributed sip ddos defense with p4. In: **2019 IEEE Wireless Communications and Networking Conference (WCNC)**. [S.l.: s.n.], 2019. p. 1–8.

FRIDAY, K. *et al.* Towards a unified in-network ddos detection and mitigation strategy. In: **2020 6th IEEE Conference on Network Softwarization (NetSoft)**. [S.l.: s.n.], 2020. p. 218–226.

GONZÁLEZ, L. A. Q. *et al.* **BUNGEE/BUNGEE-ML Repository on GitHub.** 2022. Disponível em: <https://github.com/andreyqg/ddosmitigation>.

GONZáLEZ, L. A. Q. *et al.* Bungee: An adaptive pushback mechanism for ddos detection and mitigation in p4 data planes. In: **2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)**. [S.l.: s.n.], 2021. p. 393–401.

HAMEED, S.; KHAN, H. A. Sdn based collaborative scheme for mitigation of ddos attacks. **Future Internet**, v. 10, n. 3, 2018. ISSN 1999-5903. Disponível em: <https://www.mdpi.com/1999-5903/10/3/23>.

HAUSER, F. *et al.* A survey on data plane programming with p4: Fundamentals, advances, and applied research. **ArXiv**, abs/2101.10632, 2021.

HONG, G.-C.; LEE, C.-N.; LEE, M.-F. Dynamic threshold for ddos mitigation in sdn environment. In: **2019 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)**. [S.l.: s.n.], 2019. p. 1–7.

ILHA, A. d. S. *et al.* Euclid: A fully in-network, p4-based approach for real-time ddos attack detection and mitigation. **IEEE Transactions on Network and Service Management**, p. 1–1, 2020.

ILLAVARASON, P.; SUNDARAM, B. K. A study of intrusion detection system using machine learning classification algorithm based on different feature selection approach. In: **2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**. [S.l.: s.n.], 2019. p. 295–299.

KALJIC, E.; MARIC, A.; NJEMCEVIC, P. Dos attack mitigation in sdn networks using a deeply programmable packet-switching node based on a hybrid fpga/cpu data plane architecture. In: **2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)**. [S.l.: s.n.], 2019. p. 1–6.

Kalkan, K.; Gür, G.; Alagöz, F. Filtering-based defense mechanisms against ddos attacks: A survey. **IEEE Systems Journal**, v. 11, n. 4, p. 2761–2773, Dec 2017.

Kamboj, P. *et al.* Detection techniques of ddos attacks: A survey. In: **2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON)**. [S.l.: s.n.], 2017. p. 675–679.

KFOURY, E. F.; CRICHIGNO, J.; BOU-HARB, E. An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends. **IEEE Access**, v. 9, p. 87094–87155, 2021.

KOUSAR, H. *et al.* Detection of ddos attacks in software defined network using decision tree. In: **2021 10th IEEE International Conference on Communication Systems and Network Technologies (CSNT)**. [S.l.: s.n.], 2021. p. 783–788.

Kreutz, D. *et al.* Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, v. 103, n. 1, p. 14–76, Jan 2015.

KUPREEV O. BADOVSKAYA, E. G. A. **DDoS attacks in Q2 2020**. 2020. Disponível em: <https://securelist.com/ddos-attacks-in-q2-2020/98077/>.

Lapolli, C.; Adilson Marques, J.; Gaspary, L. P. Offloading real-time ddos attack detection to programmable data planes. In: **2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [S.l.: s.n.], 2019. p. 19–27.

Li, G. *et al.* Nethcf: Enabling line-rate and adaptive spoofed ip traffic filtering. In: **2019 IEEE 27th International Conference on Network Protocols (ICNP)**. [S.l.: s.n.], 2019. p. 1–12.

MACíAS, S. G.; GASPARY, L. P.; BOTERO, J. F. Oracle: An architecture for collaboration of data and control planes to detect ddos attacks. In: **2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)**. [S.l.: s.n.], 2021. p. 962–967.

MARNERIDES, A. *et al.* Multi-level network resilience: Traffic analysis, anomaly detection and simulation. **ICTACT Journal on Communication Technology, Special Issue on Next Generation Wireless Networks and Applications**, v. 2, 2011.

MCKEOWN, N. *et al.* OpenFlow: Enabling Innovation in Campus Networks. **ACM SIGCOMM Conference on Internet Measurement**, ACM, New York, NY, USA, v. 38, n. 2, p. 69–74, mar. 2008.

MI, Y.; WANG, A. Ml-pushback: Machine learning based pushback defense against ddos. In: **Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies**. New York, NY, USA: Association for Computing Machinery, 2019. (CoNEXT '19 Companion), p. 80–81. ISBN 9781450370066. Disponível em: <https://doi.org/10.1145/3360468.3368188>.

MUSUMECI, F. *et al.* Machine-learning-assisted ddos attack detection with p4 language. In: **ICC 2020 - 2020 IEEE International Conference on Communications (ICC)**. [S.l.: s.n.], 2020. p. 1–6.

NADIM, T. U.; FOYSAL. Towards autonomic entropy based approach for ddos attack detection and mitigation using software defined networking. In: **2021 International Conference on Automation, Control and Mechatronics for Industry 4.0 (ACMI)**. [S.l.: s.n.], 2021. p. 1–5.

NARAYANAN, N.; SANKARAN, G. C.; SIVALINGAM, K. M. Mitigation of security attacks in the sdn data plane using p4-enabled switches. In: **2019 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)**. [S.l.: s.n.], 2019. p. 1–6.

P4 LANGUAGEM CONSORTIUM. **Behavioral Model (BMv2)**. 2014. [Online; accessed 03-November-2019]. Disponível em: <https://github.com/p4lang/behavioral-model>.

PAOLUCCI, F. *et al.* P4 edge node enabling stateful traffic engineering and cyber security. **IEEE/OSA Journal of Optical Communications and Networking**, v. 11, n. 1, p. A84–A95, Jan 2019. ISSN 1943-0620.

PORTER, J. **Telegram blames China for 'powerful DDoS attack' during Hong Kong protests**. 2019. Disponível em: <https://www.theverge.com/2019/6/13/18677282/telegram-ddos-attack-china-hong-kong-protest-pavel-durov-state-actor-sized-cyberattack>.

RAGHUNATH, K.; KRISHNAN, P. Towards a secure sdn architecture. In: **2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)**. [S.l.: s.n.], 2018. p. 1–7.

RANGEL, J.; VESSUM, S. van. **Amazon study: Every 100ms in Added Page Load Time Cost 1% in Revenue**. 2021. [Online; accessed 24-September-2021]. Disponível em: <https://www.contentkingapp.com/academy/page-speed-resources/faq/amazon-page-speed-study/>.

SANGODOYIN, A. O. *et al.* Detection and classification of ddos flooding attacks on software-defined networks: A case study for the application of machine learning. **IEEE Access**, v. 9, p. 122495–122508, 2021.

Santos da Silva, A. *et al.* Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn. In: **NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium**. [S.l.: s.n.], 2016. p. 27–35.

SHARAFALDIN, I. *et al.* Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. In: **2019 International Carnahan Conference on Security Technology (ICCST)**. [S.l.: s.n.], 2019. p. 1–8.

SIVARAMAN, V. *et al.* Heavy-hitter detection entirely in the data plane. In: **Proceedings of the Symposium on SDN Research**. New York, NY, USA: Association for Computing Machinery, 2017. (SOSR '17), p. 164–176. ISBN 9781450349475. Disponível em: <https://doi.org/10.1145/3050220.3063772>.

TENNENHOUSE, D. *et al.* A survey of active network research. **Communications Magazine, IEEE**, v. 35, p. 80 – 86, 02 1997.

VARALAKSHMI, I.; THENMOZHI, M.; SASI, R. Detection of distributed denial of service attack in an internet of things environment -a review. In: **2021 International Conference on System, Computation, Automation and Networking (ICSCAN)**. [S.l.: s.n.], 2021. p. 1–6.

WANG, Z.; XIAO, X.; RAJASEKARAN, S. Novel and efficient randomized algorithms for feature selection. **Big Data Mining and Analytics**, v. 3, n. 3, p. 208–224, 2020.

WEHBI, K. *et al.* A survey on machine learning based detection on ddos attacks for iot systems. In: **2019 SoutheastCon**. [S.l.: s.n.], 2019. p. 1–6.

XIA, W. *et al.* A survey on software-defined networking. **IEEE Communications Surveys & Tutorials**, IEEE, v. 17, n. 1, p. 27–51, 2015.

YAEGASHI, R.; HISANO, D.; NAKAYAMA, Y. Light-weight ddos mitigation at network edge with limited resources. In: **2021 IEEE 18th Annual Consumer Communications Networking Conference (CCNC)**. [S.l.: s.n.], 2021. p. 1–6.

ZHANG, M. *et al.* Poseidon: Mitigating volumetric ddos attacks with programmable switches. In: **NDSS**. [S.l.: s.n.], 2020.

Zhang, M. *et al.* On multi-point, in-network filtering of distributed denial-of-service traffic. In: **2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)**. [S.l.: s.n.], 2019. p. 180–188.

APPENDIXA — SUMMARY IN PORTUGUESE

As infraestruturas de rede e servidores são alvos de diversos tipos de ataques diariamente. Entre os tipos mais comuns e devastadores, encontram-se os ataques distribuídos de negação de serviço (DDoS - Distributed Denial of Service), que visam à exaustão de recursos e impactam diretamente na disponibilidade de serviços. Um ataque DDoS geralmente é executado enviando-se tráfego indesejado em massa para consumir completamente os recursos do destinatário, impossibilitando-o de oferecer seus serviços a usuários legítimos. Embora o problema venha sendo investigado há muitos anos, as propostas existentes ainda são bem limitadas em detectar e mitigar ataques DDoS rapidamente, mantendo alta acurácia e repelindo tráfego malicioso para longe da vítima (e, com isso, economizando recursos de rede).

Na última década, com o desenvolvimento e a crescente adoção de redes programáveis e, mais especificamente, de planos de dados programáveis, uma evolução passa a ser possível nos mecanismos de defesa de rede utilizando-se dos próprios dispositivos de encaminhamento de pacotes. Esses passam a ser dispositivos com capacidade para executar localmente processos mais avançados, com capacidade para contribuir na tarefa de defesa contra diferentes tipos de ataques.

Visando a abordar o referido problema a luz de avanços introduzidos por esses novos aparatos tecnológicos, nesta dissertação apresenta-se BUNGEE, um mecanismo de *pushback* colaborativo para detecção e mitigação de ataques DDoS que é executado inteiramente no plano de dados. O mecanismo é capaz de, localmente em um determinado *switch*, monitorar o tráfego e identificar endereços IP suspeitos (por meio da análise contínua de entropia de endereços origem e destino). Esses endereços suspeitos vão sendo gradativamente informados a outros *switches* na direção da(s) fonte(s) dos ataques. À medida que vão sendo informados, esses dispositivos impõem uma estratégia de filtragem para parar o ataque. O efeito prático do mecanismo proposto é barrar o ataque o mais longe possível da vítima e o mais próximo possível da(s) fonte(s). Resultados obtidos com BUNGEE demonstram a eficiência do mecanismo (em tempo de detecção) e a economia de recursos de rede que, sem o BUNGEE, seriam desperdiçados.

Em um esforço para projetar um mecanismo de detecção e mitigação rápido e, ao mesmo tempo, altamente preciso, propôs-se BUNGEE-ML, uma evolução do BUNGEE. BUNGEE-ML constitui-se em um mecanismo híbrido que combina o rápido processamento do plano de dados e a alta capacidade e inteligência potencial do plano de controle. BUNGEE-ML executa mecanismos rápidos e leves exclusivamente no plano de dados para monitorar continuamente o tráfego e detectar anomalias na rede, ao mesmo tempo em que reage rapidamente a esses comportamentos. A reação consiste em realizar *pushback* e filtragem de tráfego de origens suspeitas, processo que se denomina coordenação horizontal. Além disso, BUNGEE-ML fornece a modelos de aprendizado de máquina (executando no plano de controle) estatísticas históricas a fim de que se possa realizar uma análise de tráfego mais detalhada e refinar as listas de suspeitos (nos dispositivos que executam filtragem). Na dissertação, refere-se a esse processo como de cooperação vertical (entre o plano de dados e controle).

Para avaliar o mecanismo proposto, projetou-se e desenvolveu-se uma implementação prototípica usando-se a linguagem P4. Utilizou-se o emulador Mininet para instanciar cenários de avaliação e empregou-se *datasets* realísticos disponíveis publicamente para executar campanhas de ataques. Os resultados obtidos demonstram que BUNGEE-ML aprimora o sistema anterior (BUNGEE) ao resultar em índices comparativamente mais elevados de acurácia ao mesmo tempo que preserva característica de operação em *line rate*. Como consequência, o uso de *pushback* permite economizar ainda mais largura de banda da rede, mitigando o tráfego não legítimo corretamente e mais próximo de sua(s) fonte(s). Em relação a custos, BUNGEE-ML apresenta um baixo consumo de recursos e não sobrecarrega o canal de comunicação entre os dispositivos de rede.