جامعة الحسن الأول
UNIVERSITÉ HASSAN 1ER

Université Hassan 1er
Centre d'Études Doctorales

كلية العلوم و التقنيات سطات
FACULTÉ DES SCIENCES ET TECHNIQUES SETTAT

Faculté des Sciences et Techniques

Settat

THÈSE DEDOCTORAT

Pour l'obtention de grade de Docteur en Sciences et Techniques
Formation Doctorale: **Mathématiques, Informatique et Applications**

Spécialité:**Informatique**
Sous le thème

Cloud Computing Security Contribution based an Enhanced Software Defined Network

Présentée par :

Safaa MAHRACH

Soutenue le: 24 Juin 2021
A la Faculté des Sciences et Techniques de Settat devant le jury composé de :

Pr. Abderrahim Marzouk	**PES**	**FST de Settat**	**Président**
Pr. Mohammed Boulmalf	**PES**	**Université Internationale de Rabat**	**Rapporteur**
Pr. Abdellah Najid	**PES**	**Institut des Postes et Télécommunications, Rabat**	**Rapporteur**
Pr. Ahmed Mouhsen	**PES**	**FST de Settat**	**Rapporteur**
Pr. Said El Kafhali	**PH**	**FST de Settat**	**Examinateur**
Pr. Abdelkrim Haqiq	**PES**	**FST de Settat**	**Directeur de thèse**

Année Universitaire: 2020/2021

Acknowledgements

First of all, I would to thank Allah Subhanahu wa-Taala for the strengths and blessing in completing this thesis. My deepest and greatest gratitude to my dear parents for their endless love, prayers and encouragements. Without their constant and unconditional love, help and support, I could not have completed this thesis. I ask Allah to heal them.

This dissertation summarizes the research works conducted in the Computer, Networks, Mobility and Modeling laboratory (IR2M) of the Department of Mathematics and Computer Science at the Faculty of Sciences and Techniques of Settat.

Special appreciation goes to my supervisor, Dr. Abdelkrim Haqiq, Professor at the Faculty of Sciences and Techniques, Settat, for his guidance, support and encouragement throughout the research process. Thanks also to Dr. Dijiang Huang, Professor at Arizona State University, Tempe, AZ, USA, for his guidance and technical discussions that further enriched this thesis. I would also like to thank Professor Nate Foster and the members of the P4 Language Consortium for their help and support in performing experiments with the new P4 programing language.

I would like to take the opportunity to express my gratitude to Dr. Abderrahim Marzouk, Professor at the Faculty of Sciences and Techniques, Settat, to have kindly agreed to chair the jury of my defense. I am also grateful to my doctoral committee members, Dr. Mohammed Boulmalef, Director of the Computer Science School (ESIN) at the International University of Rabat, Dr. Abdellah Najid, Professor at the Institute of Posts and Telecommunications, Rabat, and Dr. Ahmed Mouhsen, Professor at the Faculty of Sciences and Techniques, Settat, for spending their precious time on my committee and providing insightful suggestions on the thesis. I would also like to thank Dr. Said El Kafhali, Professor at the Faculty of Sciences and Techniques, Settat, for having accepted to be an examiner for my thesis defense. I consider his presence among the jury as a testimony of great value and I am pleased to express my deep gratitude.

I warmly thank my sisters, my brothers, my friends and all the members of my family for their encouragement and moral support that allowed me to elaborate this thesis in right conditions.

Abstract

High flexibility and scalability with reduced infrastructure cost let cloud technology widely used in the public and businesses. As cloud adoption grows, the service provider and businesses need to build large data centers distributed across the globe to ensure rapid access to global tenants. This requires a scalable, flexible, and programmable cloud network infrastructure to quickly construct, modify, and provision autoscale networks.

With the emerging Software Defined Network (SDN) technology, researchers may create high-level control programs to describe the behavior of networks used for cloud management. System networks which adopt both SDN and cloud computing technologies have many advantages over traditional networks. However, new security concerns and in particular new trends of DDoS attacks have been introduced during the integration of SDN and cloud computing.

In this context, this thesis aims to protect the SDN-based cloud environment against such attacks. New security solutions have been developed to achieve this end. The first solution designs an active defensive mechanism that enables the data plane to prevent and mitigate DDOS attacks and particularly SYN flooding attacks in the cloud environment. The second one develops a lightweight and convenient DDoS mitigation system for protecting the SDN architecture and ensure a secure and resilient SDN-based environment. The third proposal concerns the design of an efficient and a secure SDN-based Openstack System Architecture.

Our proposed solutions take advantage of switch programmability, distributed packet processing, and centralized SDN control, to offer an active defense mechanisms against DDoS flood attacks in SDN-based cloud environment.The simulation results indicate that the proposed defense mechanisms may efficiently tackle the DDoS flooding attacks in the SDN architecture as well as in the downstream servers.

Keywords: Cloud Computing, Software Defined Network (SDN), Programmable data plane, Network Security, DDoS flooding attacks, DDoS mitigation, Network Intrusion Detection System, P4 programming language.

Résumé

Une flexibilité et une évolutivité élevées avec un coût d'infrastructure réduit permettent à la technologie cloud d'être largement utilisée dans le grand public et les entreprises. À mesure que l'adoption du cloud se développe, le fournisseur de services et les entreprises doivent construire de grands centres de données répartis dans le monde entier pour garantir un accès rapide aux locataires mondiaux. Cela nécessite une infrastructure réseau scalable, flexible et programmable pour construire, modifier et provisionner rapidement des réseaux auto-scalable.

Avec la nouvelle technologie SDN (Software Defined Network), les chercheurs peuvent créer des programmes de contrôle de haut niveau pour décrire le comportement de l'infrastructure réseau cloud. Les systèmes qui adoptent les technologies SDN et cloud computing présentent de nombreux avantages par rapport aux systèmes traditionnels. Cependant, de nouveaux problèmes de sécurité et en particulier de nouvelles tendances d'attaques DDoS ont été introduites dans l'environnement cloud basé sur le SDN.

Dans ce contexte, cette thèse vise à adresser les attaques DDoS qui visent à nuire l'évolutivité et la disponibilité de l'environnement cloud SDN. De nouvelles solutions de sécurité ont été développées pour atteindre cet objectif. La première solution conçoit un mécanisme de défense actif qui permet au plan de données SDN de prévenir et d'atténuer les attaques DDOS et en particulier les attaques par inondation SYN dans l'environnement cloud. La deuxième développe une solution d'atténuation DDoS légère et pratique pour protéger l'architecture SDN et garantir un environnement SDN sécurisé et résilient. La troisième proposition concerne la conception d'une architecture Openstack efficace et sécurisée basée sur le SDN.

Nos solutions proposées tirent parti de la programmabilité des switches, du traitement distribué des paquets et du contrôle SDN centralisé pour offrir des mécanismes de défense actifs contre les attaques par inondation DDoS dans un environnement cloud basé sur le SDN. Les résultats des simulations indiquent que les mécanismes de défense proposés peuvent lutter efficacement contre les attaques DDoS dans l'architecture SDN ainsi que dans les serveurs cloud.

Mots-clés: Cloud computing, Réseaux définis logiciel (SDN), Plan de données programmable, Sécurité réseau, Attaques par inondation DDoS, Atténuation DDoS, Système de détection d'intrusion réseau, Langage de programmation P4.

Contents

List of Abreviations

API	Application Programming Interface
BPF	Berkeley Packet Filter.
CUSUM	Cumulative SUM.
DDoS	Distributed Denial of Service.
eBPF	Extended Berkeley Packet Filter.
IaaS	Infrastructure as a Service.
INT	In-band Network Telemetry.
ICMP	Internet Control Message Protocol.
IoT	Internet of Things.
KVM	Kernel-based Virtual Machine.
NBI	North-Bound Interface.
NIDPS	Network Intrusion Detection and Prevention System.
NFV	Network Functions Virtualization.
NaaS	Networking-as-a-Service.
NAT	Network address translation.
OSI	Open Systems Interconnection.
OvS	Open virtual Switch.
OF	OpenFlow.
ONF	Open Networking Foundation.
ONOS	Open Network Operating System.
PaaS	Platform-as-a-Service.
PISA	Protocol Independent Switch Architecture.
P4	Programming Protocol-independent Packet Processors.
SDN	Software-Defined Networking.
SSL/TLS	Secure Sockets Layer/Transport Layer Security.
SaaS	Software as a Service.
TCP	Transmission Control Protocol.
uBPF	user-space BPF.
UDP	User Datagram Protocol.
VM	Virtual Machine.
VPC	Virtual private cloud.
VLAN	Virtual Local Area Network.
VPN	Virtual Private Network.
XDP	eXpress Data Path.

List of figures

List of tables

General Introduction

1 Introduction

Cloud Computing has been introduced as the next generation architecture of IT companies and it gives great capabilities that ensure improved productivity with minimal costs. Cloud Computing provides an enhanced level of flexibility and scalability in comparison to the traditional IT systems.

Various characteristics of cloud computing such as on-demand self-service, resource pooling, rapid elasticity, pay-as-you-go pricing model, etc. has attracted more and more enterprises to adopt the various cloud deployment models. However, the essential features of cloud computing may be the reason to cause many security challenges in cloud environment as depicted in Chapter.1. Since everything in the cloud is defined as service provided on-demand, the availability is the crucial security requirement in the cloud environment. Moreover, DDoS attacks are the main threats to interrupt the availability of cloud services[8, 9, 10].

To suffice the need of the hour, cloud needs open and programmable networks that can deal with these challenges. Software defined networking (SDN) technology simplifies the operational complexities of the traditional networking architecture by decoupling the control plane from the forwarding devices. SDN is a step towards making the network a flexible, manageable, and programmable infrastructure.

The capabilities of SDN such as software-based traffic analysis, centralized network control, simplified packet forwarding, virtualized network, etc. make it a suitable candidate to provide underneath networking support to cloud computing [11, 5].

SDN-based cloud or software-defined cloud networking is a new form of cloud networking in which SDN allows a centralized control of the network, gives a global view of the network, and provides the networking-as-a-service (NaaS) in the cloud computing environment [5], [11], [12], [13]. However, new security issues and particularly new trends of Distributed Deny of service (DDoS) attacks have been introduced over the integration of SDN and cloud computing technologies, as we can refer to the chapter.1 and [14], [11] , [15], [16]. DDoS attack may happen when an attacker forces a targeted cloud service to use excessive amounts of finite system resources like network bandwidth, memory, CPU, or disk space, which render services and computing resources unavailable. For instance, control plane saturation attack exhausts the bandwidth of the communication channel between the controller and SDN switches, as well as exhausting the switch flow-tables entries space.

The good capacities of SDN, such as software-based traffic analysis, centralized control, and dynamic network reconfiguration may greatly improve the detection and mitigation of DDoS attacks in SDN-based cloud environment [16], [17], [18],

[11]. For example, the separation of the control plane from the data plane in SDN allows IT administrators to easily perform large-scale attack and defense experiments in a real environment unlike traditional networks. Using SDN features, significant research works have been developed and proposed to defense DDoS attacks in the enterprise networking which adopt both SDN and cloud computing [16], [18], [19], [20].

Based on our study, most of the existing DDoS mitigation mechanisms are designed at the high-level SDN application plane with the involvement of the SDN controller in each operation to detect and mitigate DDoS attacks. Therefore, the communication path between the data and control planes rapidly becomes a bottleneck, which impacts the network performance and restricts its scalability and responsiveness.

To deal with these challenges, frameworks, compilers, and programming languages [21], [22], [23], [24] have been developed to take advantages of the SDN data plane with a way to perform dynamically specific operations (e.g., monitoring, detection, reaction, etc.) at the switch level.

Our research is motivated by this problem, and we ultimately intend to protect SDN-based cloud environment and to make it more resilient against DDoS attacks. New approaches and mechanisms are designed and developed to prevent and mitigate DDOS attacks in the networking environment that adopts both cloud computing and SDN technologies. The following sections formally define the research objectives and challenges, the contributions of the research, and the outline of the thesis structure.

2 Research Objectives and Challenges

The main objective of this thesis is to design a global DDoS defense system in SDN-based cloud environment. This purpose includes the following challenges:

- Survey and analysis of the capabilities of the SDN, which make it an appropriate technology for enhancing performance and network security of the cloud systems.

- Study and analysis of the new security issues introduced over the integration of SDN and cloud computing technologies.

- Review and analysis of proposed security solutions used to mitigate DDoS attacks in the SDN-based cloud environment.

- The design of an active defensive mechanism which enables the data plane to prevent and mitigate DDOS attacks and particularly SYN flooding attacks in cloud system.

- The design of a lightweight and practical DDoS mitigation mechanism for protecting the SDN architecture and ensure a secure and efficient SDN-based networking environment.

- The design of an efficient and secure SDN-based Cloud System Architecture.

3 Research Contributions

To achieve the planned purposes, the following contributions have been performed.

- The first contribution proposes an active defensive mechanism which enables the data plane to prevent and mitigate DDOS attacks and particularly SYN flooding attacks in cloud system. The proposed mechanism activates the data plane with customized statistical information databases, traffic anomaly detection algorithm and SYN flooding defense techniques. These extensions enable the switch to prevent the overwhelming traffic attacks, mitigate the SYN flooding threats and then deploy adaptive countermeasures. Our system takes advantage of the switch programmability (i.e., using P4 language), distributed packet processing, and centralized SDN control, to offer an active defense mechanism against SYN flood attacks in cloud environments. This contribution has been the subject of the following publications:

  [1] Safaa Mahrach, Oussama Mjihil, and Abdelkrim Haqiq. "Scalable and Dynamic Network Intrusion Detection and Prevention System". In the Proceedings of the International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2017), held in Marrakesh, Morocco, 11-13 December, 2017. Published on the book of Advances in Intelligent Systems and Computing, Vol. 735, pp. 318-328, Springer, 2017.

  [2] Safaa Mahrach, Iman EL MIR, Abdelkrim HAQIQ, and Dijiang Huang. "SDN-based SYN Flooding Defense in Cloud". Journal of Information Assurance and Security, Vol. 13, pp. 30-39, 2018.

- The second contribution aims to design a lightweight and practical mitigation mechanism to protect SDN architecture against DDoS flooding threats and ensure a secure and efficient SDN-based networking environment. Our proposal extends the Data Plane (DP) with a classification and mitigation module to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures. The simulation results indicate that the proposed defending mechanism may efficiently tackle the DDoS flood attacks in the SDN architecture and also in the downstream servers. This contribution has been the subject of the following publications:

  [3] Safaa Mahrach and Abdelkrim Haqiq. "DDoS defense in SDN-based Cyber-Physical Cloud". Cybersecurity and Privacy in Cyber Physical Systems, pp. 133-158, 2019.

  [4] Safaa Mahrach and Abdelkrim Haqiq. "DDoS Flooding Attack Mitigation in Software Defined Networks". International Journal of Advanced Computer Science and Applications, Vol. 11, pp. 693-700, 2020.

- The third contribution designs an efficient SDN-based Openstack System Architecture. The proposed design architecture uses the Openstack opensource cloud platform and the ONOS SDN controller. The proposal architecture aims to activate a high-performance and reconfigurable datapath based on the combination of the Programming Protocol-independent Packet Processors

(P4), the P4 Runtime API, and the Open vSwitch software switch. To protect the SDN-based Openstack platform, we opt to implement the proposed DDoS mitigation mechanism of the previous contribution which aims to protect the SDN components from DDoS flood attacks and achieve an efficient networking system. As a result, we get a secure and resilient SDN-based Openstack platform that can resist DDoS flooding attacks. This contribution has been the subject of the following communication:

[5] Safaa Mahrach and Abdelkrim Haqiq. "DDoS Attack and Defense in SDN-based Cloud". In the Proceedings of the International Symposium on Ubiquitous Networking (UNet'21), held online, 19-21 May, 2021. Springer LNCS, 2021.

4 Thesis structure

This thesis is organized in four chapters, in addition to a general introduction and a general conclusion:

- In the first chapter, we attempt to give a general overview of cloud computing, its architecture, service models, deployment models, and characteristics. Then we presented SDN architecture and we discussed their features which make it an appropriate technology for cloud system. Afterward, we analyzed the new security issues introduced over the integration of SDN and cloud computing technologies. Subsequently, we examined how the cloud and SDN characteristics make them more vulnerable to DDOS attacks. At the end of this chapter, we reviewed and analyzed the recent proposed research approaches and we presented an interesting state of the art on DDoS detection and mitigation in the SDN-based cloud environment

- In the second chapter, we present the first proposed defensive mechanism which enables the data plane to prevent and mitigate DDOS attacks and particularly SYN flooding attacks in cloud systems. The proposed mechanism extends the data plane with customized stateful databases, the cumulative sum (CUSUM) traffic anomaly detection algorithm, and SYN flooding defense methods. These extensions enable the switch to prevent the overwhelming traffic attacks, mitigate the SYN flooding threats and then deploy adaptive countermeasures. Our system takes advantage of the switch programmability (i.e., using P4 language), distributed packet processing, and centralized SDN control, to offer an active defense mechanism against SYN flood attacks in cloud environment.

- In the third chapter, we present the second contribution. The design and implementation of a lightweight and practical mitigation mechanism to protect the centralized SDN controller and the SDN data plane from SYN flood attack, and achieve an efficient networking system that can resists against DDoS flooding attack. Our proposed solution exploits the high processing power of data plane to perform both prevention and mitigation techniques to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures. Due to the limited memory of the data plane, we opted to use the stateless SYN cookie technique as prevention technique. For implementing the proposed SYN flood mitigation mechanism we use the P4 programming language [4] and bmv2 (behavioral-model) software switch in Mininet emulator.

- In the fourth chapter, we present the third contribution. The design of an efficient SDN-based Openstack System Architecture. The proposed design architecture uses the Openstack opensource cloud platform and the ONOS SDN controller. The proposal architecture aims to activate a high-performance and reconfigurable datapath based on the combination of the Programming Protocol-independent Packet Processors (P4), the P4 Runtime API, and the Open vSwitch software switch. To protect the SDN-based Openstack platform, we opt to implement the proposed DDoS mitigation mechanism of the previous contribution which aims to protect the SDN components from DDoS flood attacks and achieve an efficient networking system. As a result, we get a secure and resilient SDN-based Openstack platform that can resist DDoS flooding attacks.

Chapter 1

Survey on SDN-based Cloud and Security

1.1 Introduction

Cloud computing is growing rapidly in industry and academia due to its important characteristics including resource pooling, on-demand self-service, multi-tenancy, elasticity, and so on. Cloud technologies have significant benefits for the economy while reducing capital expenditure (CapEx) and operational expenditure (OpEx) [25]. Virtualization is a key for cloud computing since it allows abstraction of the underlying hardware platform (e.g., computing, network, and storage) for sharing with other tenants, and isolation of services and applications running on the same physical server [26]. Like virtualization, Software Defined Networking (SDN) is a new paradigm widely deployed in cloud computing, in which groups of switches and network devices are deployed over the large network as shared virtual resources. SDN concept is focused on a dynamic and automatic management of network services from a high abstraction level by separating the control and data planes.

SDN-based cloud or software-defined cloud networking is a new form of cloud networking in which SDN allows a centralized control of the network, gives a global view of the network, and provides the networking-as-a-service (NaaS) in the cloud computing environment [5, 11, 12, 13]. However, new security issues and particularly new trends of DDoS attacks have been introduced over the integration of SDN and cloud computing technologies [14, 11, 15, 16].

The good capacities of SDN, such as software-based traffic analysis, centralized control, and dynamic network reconfiguration may greatly enhance the detection and mitigation of DDoS attacks in SDN-based cloud environment[16, 17, 18, 11]. Using SDN features, significant research works have been developed and proposed to defense DDoS attacks in the enterprise networking which adopt both SDN and cloud computing [16, 18, 19, 20].

The remainder of this chapter is structured as follows: The section 1.2 represents a general overview of cloud computing and SDN technologies, their architecture, characteristics, and usage, also we present the general architecture of SDN-based cloud environment and the different types of DDoS attacks and the methods and techniques used to detect and mitigate such attacks. Additionally, we discuss security issues related to cloud computing and SDN and new security issues introduced over the integration of SDN and cloud computing technologies in section 1.3. In section 1.4, we discuss how the cloud characteristics make it more vulnerable to DDOS attacks, we discuss how SDN components can be targeted by DDoS attacks, then we examine the SDN features that improve mitigation of DDOS attack in SDN-based cloud. In section 1.5, we evaluate some of the existing SDN-based DDoS detection and mitigation solutions, and then we present the frameworks and programming languages proposed to address the limitations of the proposed solutions.

1.2 Overview

Adopting SDN technology in the cloud computing environment, to take control of the cloud network infrastructure, can simplify the operational complexities of the network and dramatically enhance the management, programming, and scalability of the cloud network. In this section, we present the cloud computing, their characteristics, architecture, service and deployment models, also we discuss the SDN architecture and their capabilities which make it an appropriate technology for the cloud networking system. In addition, we present the general architecture of SDN-based cloud environment and the different types of DDoS attacks and the methods and techniques used to detect and mitigate such threats.

1.2.1 Cloud Computing

There are a lots of definitions and metaphors of cloud computing. Below, we present few definitions of cloud computing:

"Cloud computing is the on-demand availability of computer system resources, especially data storage (cloud storage) and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet". Says Wikipedia [27].

The National Institute of Standards and Technology (NIST) [28] has defined cloud computing as "a model for enabling ubiquitous and on-demand network access to shared resources (e.g., applications, services, networks, servers, and storage) that can be quickly provisioned and released with minimal management effort or service provider interaction".

The IEEE Standards Association (IEEE-SA) defines cloud computing in two working drafts. The P2301 (Cloud Profiles) draft highlights different ecosystem of cloud such as cloud vendors, service providers, and users. P2302 (Intercloud) draft provides definition on topology, functions, and governance for cloud-to-cloud interoperability and federation.

Gartner group, forecaster of Information technology, defines cloud computing as "a style of computing in which scalable and elastic IT-enabled capabilities are delivered as a service using Internet technologies".

BSI is the Federal Cyber Security Authority in Germany has specified the following definition [29]: " Cloud computing is understood as offering, using, and billing IT services dynamically adapted to the requirements, via a network. Here, these services are only offered and used by means of defined technical interfaces and logs.

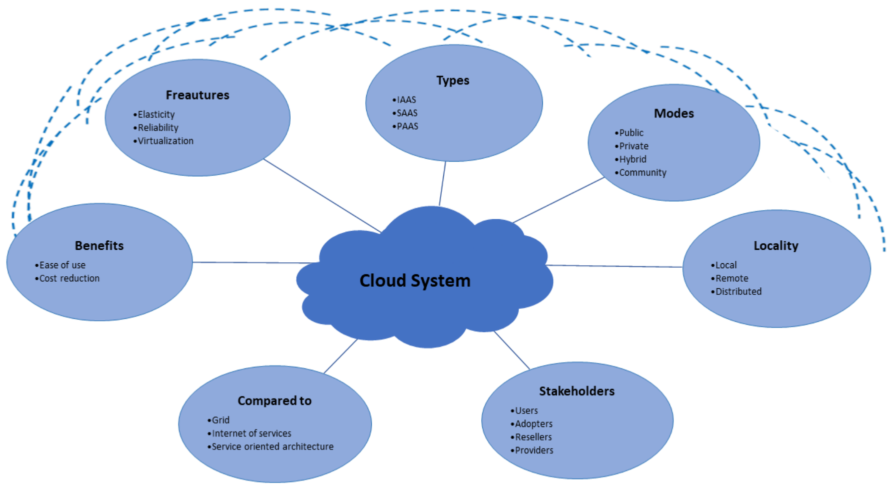


Figure 1.1 – Cloud Computing System

The range of the services offered within the cloud computing framework covers the entire spectrum of information technology and, among other things, includes infrastructure (e.g. computing power, storage space), platforms and software".

A. Cloud computing architecture layer

Cloud Computing architecture has four layers: Hardware layer, Infrastructure layer, Platform layer, and Application layer, as shown in Figure. 1.2.

- **Hardware layer:** The hardware layer manages the cloud physical resources include physical servers, switches, routers, power, and cooling systems. Practically, the hardware layer is usually implemented in date centers. This latter generally houses thousands of servers that are interconnected via switches, routers or other materials. This layer is exposed to several challenges, including hardware configuration, traffic management, power and cooling resource management, and fault-tolerance.

- **Infrastructure layer:** The infrastructure layer provides a shared computing, storage, and network resources by abstracting the physical layer using virtualization technologies. The infrastructure layer is the basic element of cloud, since many key features of cloud, such as resource pooling and scalability, are enabled using virtualization technology.

- **Platform layer:** The third layer of cloud is the platform layer which provides software and app frameworks to build easily the applications. This platform provides developers swift access to a full development and deployment environment. In combination with IaaS, PaaS provides the ability to develop, test, run, and host applications.

- **Application layer:** The application layer consists of the actual cloud services or applications. The cloud applications can leverage the automatic scaling feature to achieve better performance, availability and lower operating cost.

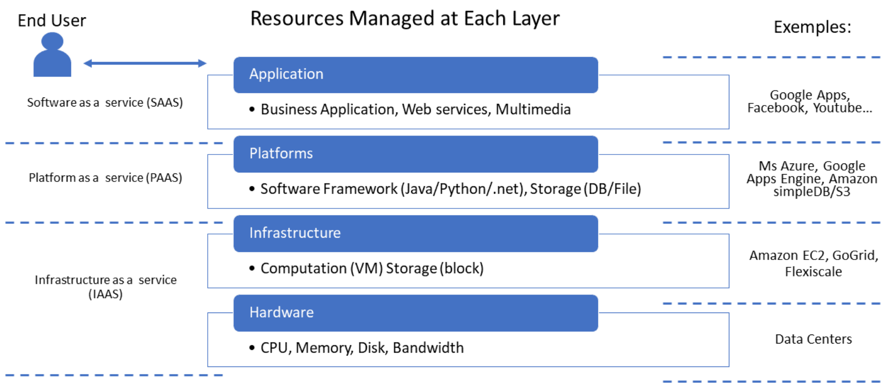

End User
Resources Managed at Each Layer
Exemples:
Software as a service (SAAS)
Application
• Business Application, Web services, Multimedia
Google Apps, Facebook, Youtube...
Platform as a service (PAAS)
Platforms
• Software Framework (Java/Python/.net), Storage (DB/File)
Ms Azure, Google Apps Engine, Amazon simpleDB/S3
Infrastructure as a service (IAAS)
Infrastructure
• Computation (VM) Storage (block)
Amazon EC2, GoGrid, Flexiscale
Hardware
• CPU, Memory, Disk, Bandwidth
Data Centers

Figure 1.2 – Cloud Architecture

The modularity of cloud architecture allows each layer to evolve separately, making it easy to improve security and scalability of the cloud.

B. Cloud computing service models

Cloud computing is offered in three different service models which known as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS).

- **Infrastructure as a Service (IaaS):** IaaS provider offers their costumers direct access to a virtual infrastructure resources including computing, storage, and networks through the Internet. The key advantage of IaaS model is that customers need to pay only for what they consume. Also, there is no need to engage experienced IT staff to manage and upgrade the infrastructure. IaaS services offered today, include Amazon EC2, Microsoft Azure, and Google Cloud Platform.

- **Platform as a Service (PaaS):** PaaS provider helps developers to focus on software app development or providing new services to customers without having to manage, update, and maintain the software platform for the app. As with IaaS, PaaS is world of services that not required to buy, configure, and manage IT hardware, but instead the consumers pay only for their needed virtual server resources. More than that, the software platform which may host the costumer's applications is provided. PaaS services offered today, include Amazon EC2, Oracle Cloud, and Google Cloud Platform.

- **Software as a Service (SaaS):** SaaS refers to providing on-demand applications via Internet without requiring costumers to actually own or install those applications. Applications, such as web-based email, a business app, or even a word processor that run entirely in the cloud. Key advantages of SaaS are accessibility, updates and compatibility, and operational management. SaaS services offered today, include Amazon Web services, Google Cloud G suite, and Cisco WebEx.

C. Cloud Computing deployment models

Cloud computing is presented in four different deployment models named Public Cloud, Private Cloud, Hybrid Cloud, and Virtual Private Cloud.

- **Public Cloud:** Public cloud model offers tenants quick access to computing resources without a significant initial cost. With public cloud, tenants may simply buy virtualized compute, storage, and networking services through Internet. Key benefits of public cloud, users need to pay only for what they use, also they can scale easily since they can simply extend the cloud's capacity as the requirements increase. Their is no need to develop and maintain the software. The top cloud service providers today include Microsoft Azure, Amazon Web Services, Google Cloud, Alibaba Cloud, and IBM.

- **Private Cloud:** Private cloud model belongs to a specific company. It is hosted in the client's data center and maintained by its IT team, which demands a significant expenditure capital. The private model allows tenants to control how data is stored and shared and to customise their infrastructures in accordance with their needs. This is the best cloud model if security is the client concern, since the client can direct data governance, guarantee conformity with any regulations, and protect valued intellectual property. In addition, the private cloud gives clients on-demand data availability, assuring accuracy and support for mission-critical operations. Several public cloud service providers, including Google, Amazon, IBM, Cisco, and Red Hat, also provide private cloud.

- **Hybrid Cloud:** Hybrid cloud is a combination of two or more cloud deployment models (public, private, community, VPC, dedicated servers). This model helps customers seamlessly scale up services between the on-premises data center and the public cloud. As an example, a company can balance its loads by storing critical workloads on a private cloud and less sensitive ones on a public cloud. Key advantages of this model are enhanced security and privacy, improved scalability and flexibility, and reasonable price. Microsoft Azure Stack, AWS Outposts, Google Cloud Anthos, Oracle Cloud, IBM.

- **Virtual Private Cloud:** Virtual private cloud (VPC) is a secure, isolated private cloud contained within a public cloud. VPC logically isolates customer's computing, storage, network resources from the other customer's resources available in the public cloud. This isolation may be accomplished by using some or all of the following technologies: subnets, VLAN, encryption and tunneling with VPN, and NAT. VPC enables tenants to take advantage of the benefits of the private cloud, such as more granular control over virtual networks and a secure isolated environment for sensitive data and applications, while leveraging the scalability and convenience of public cloud resources. Most leading public IaaS providers, including Amazon Web Services (AWS), Microsoft Azure and Google, provide VPC and virtual network services.

D. Cloud computing characteristics

Below the key characteristics that differentiate cloud computing from traditional, on-premises data center architectures.

- **On-demand self-service:** The Cloud services don't need any human interaction, consumers themselves may provision automatically resources, such as Web applications, server time, network access, storage etc, as required. With this characteristic, client can also monitor and manage the computing resources.
- **Large network access:** Customers may access resources of cloud through the public Internet connection all the time and from anywhere using various types of devices, such as laptops, smart phones, tablets etc.
- **Resource pooling:** The resource pooling feature rely on multi-tenant model and virtualization technology to serve multiple clients at the same time. Physical and virtual resources are shared into the cloud to serve several customers. In general, the customer has no control or knowledge over the location of the provided resources.
- **Scalability and rapid elasticity:** Resource pooling enables both cloud providers and customers to scale up or down quickly. Cloud providers can add new servers and network devices to cloud with minor modifications to cloud infrastructure and software. Also, customers can rapidly and elastically provision and release computing, storage and networking resources as needed. With cloud computing scalability, there is less capital expenditure on the cloud customer side.
- **Measured service:** Cloud systems automatically control and optimize the use of resources by leveraging measurement capability at a certain level of abstraction appropriate to the type of service (for example, storage, processing, bandwidth, and active user accounts). Resource usage can be monitored, controlled, and reported, ensuring transparency for both the provider and consumer of the utilized service.
- **Multi-tenancy:** Multi-tenant model is a vital element of cloud computing. Multi-tenancy allows customers to share the same applications or the same physical infrastructure while keeping isolation and security over their data. Within the cloud infrastructure, tenants have an isolated space for storing their projects and data which can only be accessed with the cloud provider's permissions. Compared to single-tenant, multi-tenant allows better use of resources while guaranteeing the privacy and security of the tenant's information. By sharing resources among multiple tenants, the cloud provider may offer services at a much lower cost.

E. Virtualization technology in cloud computing

Virtualization is the process of creating a virtual instance of a hardware resource. The virtual environment created can be a single instance or a combination of operating systems, software applications, computing, networks, storage devices, etc... These virtual environments are then shared among multiple cloud users using numerous services such as infrastructure, software, and platforms. Without building their own infrastructure, tenants have easy access to cloud servers, storage, and network resources using virtualization technology. In addition, virtualization allows users to access huge amounts of storage as needed, and scale up and down as needed, without the need for their own storage.

There are various types of virtualization in cloud computing:

- **Hardware virtualization:** It consists of a hypervisor which is capable of creating virtual machines and then managing and allocating resources (e.g., CPU, memory, and other hardware resources) to them. Hypervisor can either be installed directly on the hardware, or work as a layer installed on the operating system.

- **Network virtualization:** It refers to the process of combining hardware and software network resources into a single software-based network. This creates a virtual network that enables administrative control over all the network resources. This is done through a central virtual network management system such as SDN controller. Additionally, network virtualization enables cloud vendors to provide multiple virtual networks with each has a separate control and data plan.

- **Application virtualization:** It allows cloud users to run software applications from a remote server. This means users do not need to have enough storage space in their machines to install and run an application because the app is installed on the cloud servers.

- **Operating system (OS) virtualization:** It enables cloud providers to run multiple OS in one server, each virtual OS is separate from the others and is protected. One of the main uses of OS virtualization is for testing software applications on different OS and platforms.

The main benefits of using virtualization in cloud computing are protection from system failures, ease to transfer machines or data, security, cost-effective strategy, and Smoother IT Operations.

Since virtualized infrastructure is separated into containers, even if one part of the system goes down, the rest of the system will be protected from issues, bugs, or application crashes. The easy transfer of the entire machines is possible with desktops and storage virtualization, without needing to change the physical infrastructure. Moreover, data can be easily located and transmitted between devices and servers within no time. Security is one of the important concerns in cloud virtualization, it can be achieved using virtual firewalls to prevent unauthorized access to data and cloud resources. Encryption process also helps protect data against various threats. Since all data is stored on virtual servers, this reduces waste, lowers electricity bills as well as maintenance costs. In addition, virtual networks and cloud computing can facilitate the use of centralized management processes, making it easier for businesses and IT departments to operate.

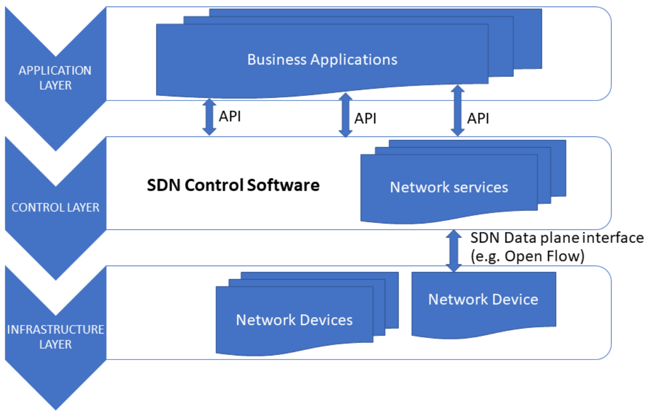


Figure 1.3 – SDN Architecture [1]

1.2.2 Software Defined Networking - SDN

SDN is seen as a key technology for improving the management and control of large-scale networks like cloud computing. SDN separates the control layer and the forwarding layer, in order to make the traditional networks more flexible, dynamic, and programmable and to allow applications and network services to directly control the abstract infrastructure [30, 31, 32]. OpenFlow was introduced as the standard communication protocol between the control plane and the network devices.

Open Networking Foundation (ONF) is a non-profit operator-led consortium that drives the transformation of network infrastructure and operators' business models [1]. According to the ONF the SDN architecture consists of three layers including the data plane, control plane, and application plane. The basic structure of SDN is represented in Figure.1.3.

- **Application plane:** It is responsible for managing business and security applications. Among the applications addressed by this layer, we cite network virtualization, intrusion prevention systems (IPS), intrusion detection systems (IDS), and firewall implementation.

- **Control plane:** It consists of an SDN controller or a set of controllers, which centrally program and control behavior of the network elements through south-bound API such as OpenFlow and NetConf [33, 34]. The controllers in the distributed environment communicate with each other through East and West-bound APIs. As the south-bound interface, the controller also offers a similar interface with the application layer called the north-bound API to extend the services of the application running in the upper layer.

- **Data plane:** It consists of network devices such as switches, routers, and wireless access point. Both virtual switches such as OpenFlow, Open vSwitch, P4, Pica8 [33, 35, 4, 36] and the physical switches coexist in this infrastructure. The main function of the data plane is simply transmit packets using the decisions (i.e., flow rules) assigned by the SDN controller.

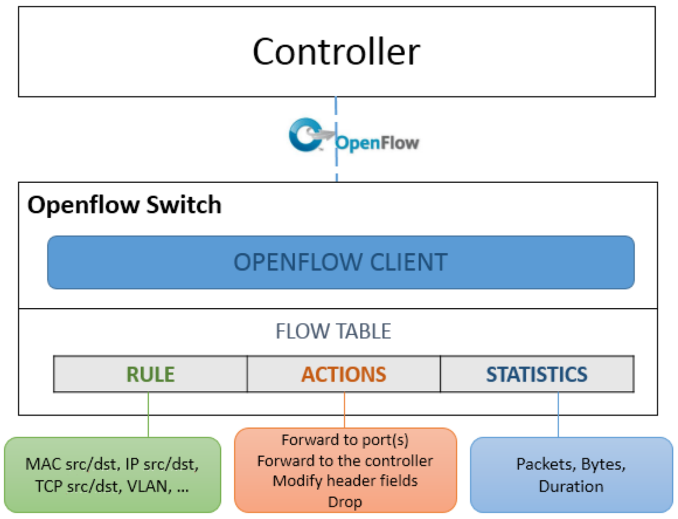


Figure 1.4 – OpenFlow Architecture [2]

A. SDN characteristics

The characteristics of SDN allow obtaining high performance and flexibility compared to traditional networks. Below, we cite some key features of SDN:

- **Centralized control:** SDN provides centralized control over the entire network using a controller or a set of controllers such as OpenDaylight, Floodlight, Maestro, NOX, POX, etc. The controller maintains a global view of the network, which appears to the application layer as a single and logical switch.

- **Programmable Configuration:** SDN allows IT managers to configure, manage, and secure all networking elements such as switches, routers, and firewalls using automated programs. As such, an entire network can be programmatically configured and dynamically optimized based on the network status.

- **Innovation:** The openness of SDN encourages further exploration, innovation, and development of the network. In contrast, conventional networks are vendor-specific and cannot be changed.

- **Low economic cost:** Centralized control and implementation of logic in switches eliminate the deployment of thousands of switches. Thus, the total cost allocated for the maintenance of switches is reduced, as well as the total cost of network equipment.

B. The standard OpenFlow protocol

The OpenFlow [2] protocol has been created and considered as the standard southbound interface between the SDN controllers and the network elements. OpenFlow interface is responsible for implementing packet-forwarding rules in the switch flow tables, which are used to handle packets in line rate.

In the OpenFlow architecture, illustrated in Fig.1.4, the OpenFlow switch contains one or more flow tables and an abstraction layer (i.e. Secure Channel) that securely communicates with the controller via the OpenFlow protocol. Flow tables consist of flow entries which determine how packets will be processed. Flow entries typically consist of:

- **Matching rules**, used to match incoming packets; match fields may contain information found in the packet header (Ethernet Src/Dst, IP Src/Dst . . .).
- **Actions**, which define how the packets should be processed.
- **Counters**, used to collect statistics for a particular flow, such as number of received packets, number of bytes and duration of the flow.

C. SDN usage in cloud

The use of SDN in cloud data centers can be presented for different purposes [5].

Virtualization: Cloud computing consists of different virtualization technologies that can allow running virtual machines (VMs) in clouds. Hypervisors running on physical hosts can virtualize the host resources such as processor, memory, and storage. In addition, SDN technology is able to virtualize network resources (e.g., switches and physical links) and rent them to divers tenants in the cloud data center. Network Function Virtualization (NFV) joins the cloud and SDN to accelerate the development of new network services with elastic scalability and automation. NFV brings agility in delivering network services by removing bottlenecks imposed by manual processes and enabling the deployment of new services on demand.

Security: Security is the major concern in cloud computing, since everything is provided as service. SDN technology was widely used to defend, prevent and detect security attacks (e.g., DDoS attacks) in cloud data centers. Due to SDN features such as centralized control, traffic analysis, and dynamic update of network rules, the system will be able to resist such attacks. Although, SDN has been widely adopted to protect cloud and other systems, it is necessary to secure the SDN controller as it is the core component that manages the entire data center network.

Energy efficiency: As cloud data centers consume huge amounts of electricity around the world, energy-saving methods have received great attention in the cloud data centers over recent years. SDN allows network elasticity in addition to the computing elasticity. If the system declares low network usage in some switches, network traffic can be consolidated into the smaller number of switches using SDN, which gives the ability to turn off unused switches.

SDN-based solutions have attracted great attention recently. Various research approaches and solutions have been developed in this field. Cziva et al. [13] developed an SDN-based framework for live Virtual Machine (VM) to make easy network-server resource management over Cloud Data Centers. The live VM migration exploits temporal network information in order to reduce the network-wide communication cost of the resulting traffic dynamics, and alleviate congestion of the high-cost. In [37] Pisharody et al. designed a security policy analysis framework to detect all potential conflicts over a distributed SDN-based cloud environment by leveraging the programmability of SDN. The detection method is extended from the

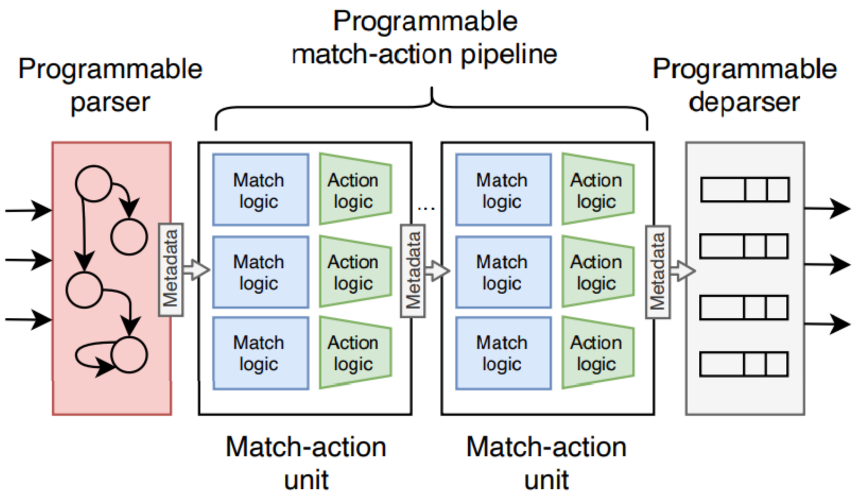


Figure 1.5 – Protocol-Independent Switch Architecture (PISA) [3]

firewall rule conflict detection techniques in traditional networks. Son et al. [12] proposed a dynamic overbooking algorithm for consolidating VMs and traffics in cloud Data centers while exploiting jointly leverages virtualization and SDN abilities. The main objective of their proposed approach is to minimize SLA violations and save energy. From the same institution Son et al. [5] presented a taxonomy of the usage of SDN in cloud computing in various aspects including energy efficiency, performance, virtualization, and security enhancement . The authors in [38] proposed a lightweight policy enforcement system, SDNKeeper, to protect and flexibly manage network resources in the SDN-based cloud environment. They developed a generic policy language for administrators to create management policies. SDNKeeper performed access control on North-Bound Interface (NBI) based on the policies defined by administrators. It used plugins, policy interpreter and permission engine, to effectively verify the legitimacy of access requests against the predefined policies.

1.2.3 Programmable Data Plane

Protocol Independent Switch Architecture (PISA) is a data plane programming model [39]. PISA devices is programmed using the high-level P4 language. PISA is based on the concept of a programmable match-action pipeline. It consists of programmable parser, programmable match-action pipeline, and programmable deparser, as depicted in figure.1.5.

- The programmable parser allows programmers to declare arbitrary headers and define their order in the packets.

- The programmable match-action pipeline consists of multiple match-action units (or stage). Each unit includes one or more match-action-tables (MATs) to match packets and execute match-specific actions. A single MAT has memory block for storing tables and registers and Arithmetic Logic Units (ALUs) with computational capabilities and stateful memories (e.g.,counters and meters). A control plane manages the matching logic by writing entries in the MATs to influence the runtime behavior.

- The programmable deparser assembles the packet headers back and serializes them for transmission.

A. Benefits of Programmable Data Plane (PDP)

- **Flexibility:** PDP allows programmer to design, test, and adopt new algorithms, protocols, and features in shorter times.
- **Top-down design:** PDP enables network equipment designers and end users to create and deploy new protocols and features without depending on the providers of the specialized packet processing ASICs to implement custom algorithms.
- **Efficient System:** with PDP, programmers can deploy only the necessary protocols, instead of the fixed-function data plane which incorporates multiple protocols that consume resources and add complexity to systems.
- **Visibility:** PDP provides better visibility into the behavior of the network. For example, In-band Network Telemetry (INT) is a framework for collecting and retrieving information from the data plane, without the intervention of the control plane.

B. Programming Protocol independent Packet Processors (P4)

Programming Protocol independent Packet Processors (P4) is a programming language for describing how packets are processed by the data plane. It's also considered as a protocol between the controller and the network devices [4]. P4 design has three main goals:

Protocol independence: P4 enables a programmable switch which can define new header formats with new field names and types.

Reconfigurability: The protocol independence and the abstract language model enable reconfigurability by changing the way of packets processing after deploying the switches.

Target independence: P4 programs can be compiled on many different types of network infrastructure such as CPUs, FPGAs, and ASICs. Instead, a compiler should take into account the switch's capabilities when turning a P4 program into a target-dependent program (i.e., used to configure the switch).

P4 program is based upon an abstract forwarding model consisting of a parser and a set of match-action table resources, divided between ingress and egress (Fig.1.6).

- **Parser:** Defines the headers present in each incoming packet.
- **Match-action tables:** As P4 specification, tables are the fundamental units of the match action pipeline. Tables define rules to perform the input fields to use, and the actions that may be applied. Actions in P4 are declared as functions (i.e., compound actions) which are built from primitive actions (i.e., basic actions).

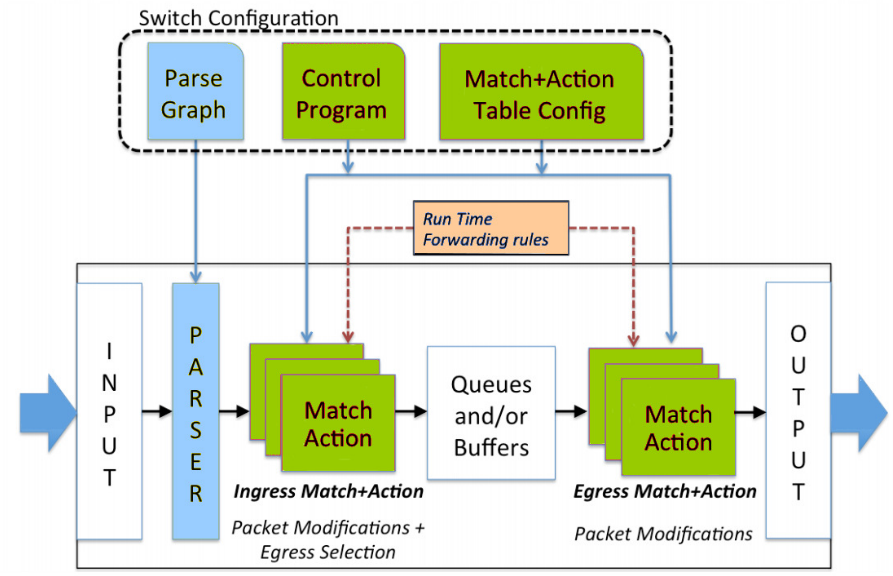


Figure 1.6 – Abstract forwarding model [4]

- **Stateful memories:** P4 maintains information across packets using stateful memories including counters, meters and registers. In this work, we use counters to perform different measurements, such as the number of packets or bytes generated by each host that can enable the detection of eventual worm propagation or Denial-of-service (DOS) attacks. Counters are categorized in two types: direct and indirect.

  ```
  Counter ip_pkts_by_dest{
      type: packets;
      direct: ip_host_table;}
  ```

  In the example above we use direct access for ip-host-table, which allocates one counter for each table entry. The current counter gives to the controller the ability to read the number of packets sent to each host.

- **Control Program:** The control program organizes the layout of tables within ingress and egress pipeline, and the packet flow through the pipeline. It maybe expressed with an imperative program (i.e., main program in P4 language) which may apply tables, call other control flow functions, or test conditions.

1.2.4 SDN-based cloud computing environment

Due to the Cloud computing characteristics including on-demand self-service, multi-tenancy, elasticity, and broadband network access, IT managers face greatest challenges [40], [41], [42]. Availability is seen as the critical security requirement in the cloud since everything in the cloud is defined as a service [42], [43]. As cloud infrastructure and applications are distributed around the world, this leads to much-anticipated scalability challenges [44], [45].

Due to the growth of cloud systems, the service provider needs to create a large and distributed number of data centers around the world to ensure rapid access to global customers. This places an increasing demand and usage of the cloud networking infrastructure [46], [47]. To address these challenges, a programmable and flexible network is needed which may be performed while separating the control decision from the network devices. In particular, SDN technology may be used to simplify the management of cloud networking while decoupling network intelligence from forwarding functions. SDN-enabled cloud data center or SD cloud networking is a new form of cloud data center networking, in which SDN technology allows centralized control and programmable configuration of the network and provides networking-as-a-service (NaaS) in cloud computing environment [5], [11], [12], [13], [38].

A comprehensive survey on SDN-enabled cloud data centers has been presented below. SDN-enabled Cloud Data Center (SDN-DC) has been proposed by Hwang et al. in [48]. SDN-DC adopts SDN features to solve the various issues of a cloud data center. They developed an SDN-DC architecture that addresses significant traffic engineering challenges including auto topology discovery, auto addressing, and auto-routing using known protocols and mechanisms. In addition, the proposed architecture supports rapid fault detection and recovery and load balancing through centralized control. Software-defined cloud computing (SD-clouds) has been proposed by Jararweh et al. [49] and Buyya et al. [50] where not only the networking but the entire infrastructure components in cloud computing are considered to be software-defined. The approach is to realize a fully automated cloud data center that can optimize configurations autonomously and dynamically. In [50] the authors proposed an optimal cloud environment by extending the virtualization concept to all resources of a data center including server virtualization, SDN, network virtualization, and virtual middleboxes. Jararweh et al. [49] attempt to take advantage of the Software-defined systems (SDSys) capabilities to handle the complexities associated with cloud computing systems. They focus on the integration of SDN, software-defined compute (SDCompute), software-defined storage (SDStorage), and software-defined security (SDSec).

A. SDN-based cloud Architecture

Figure. 1.7 indicates the architecture of the standard SDN-enabled cloud systems obtained from the literature [5, 11]. *Cloud Manager (CM)* controls the cloud tenants and resources including compute, networking, etc. CM manages the tenant requests such as VM creation and provisions cloud resources to supply the cloud services. It also controls energy-efficient resources and resource monitoring. OpenStack [51] is an open-source instance of a CM widely used to create public and private clouds. Network services are managed by *SDN controller* which communicates with the cloud manager using north-bound APIs. SDN controller is responsible for performing the essential network functions such as network monitoring, network topology discovery, dynamic network configuration, and virtual network management. Scalability is possible using multiple SDN Controllers communicating through east/west-bound APIs. *Cloud resources* include compute and networking resources. Compute Resources (CRs) are the servers that host VMs using hypervisor such as KVM,

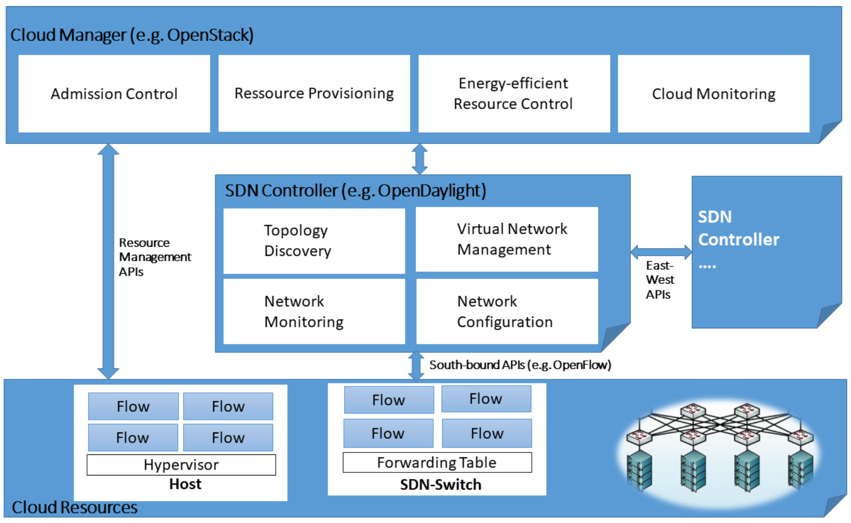


Figure 1.7 – SDN-based cloud Architecture [5]

Xenserver, Vmware, etc. CRs are controlled by the cloud manager via resource management APIs. Networking Resources (NRs) are the switches that can support multiple virtual switches using Open vSwitch (OvS). Physical and virtual switches (i.e., both virtual switches hosted in switches and in hosts) are controlled by SDN controller by updating forwarding tables in switches via South-bound APIs (e.g., OpenFlow, P4 Runtime).

1.2.5 Distributed Denial of Service attacks - DDoS

Although DDoS attacks have been known from the origins of Internet networks, DDoS threats are still increasing in today's networks which increases awareness of them.

The main objective of DDoS attack is to make network resources, compute resources or services unavailable and thus it has become impossible for legitimate users to use them. In this way, the attacker involves the control of various devices (botnet) and uses them to overload the victim with overwhelming traffic. DDoS attack is relatively easy to execute, difficult to defend, and the attacker is hardly traced back. The diagram of the DDoS attack is shown in Figure.1.8.

The attacker launches DDoS attack using botnet or multiple botnets to generate a huge amount of traffic against a victim to exhaust its resources and make it unavailable. Botnet is a large number of IP devices connected to the Internet such as PC's and servers. Botnet can also be integrated devices such as cell phones, routers, cameras, alarm systems, etc. For example, the internet of things (IoT) is a network connecting hundreds of millions of physical devices including electronics, software, vehicles, transportation, sensors, and actuators. These devices connect and exchange data, and can also be compromised and act as a botnet for DDoS attacks. The attack can deactivate the targeted server or the network which can conduct to deactivating the whole organization.

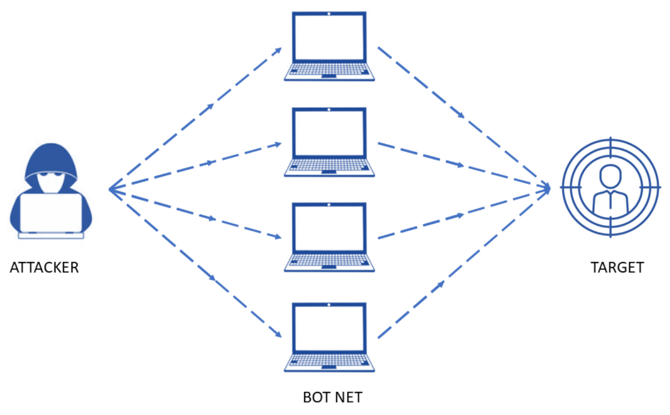


Figure 1.8 – DDoS attack diagram

A. DDoS attack types

We can classify DDoS attacks in two categories: application-layer attacks and Network layer attacks.

Application layer attack: It is done at layer 7 of the OSI model. The aim of this attack is to overload an online service, an application, or a website. Application layer attack is characterized by low and slow rates which hardly make detection of these attacks. Although known as Layer 7 attack, it can be part of attacks targeting bandwidth and network. In an application-level attack, amplification is based on processor, memory, or resource. This type of attack is measured in magnitude by requests per second (Rps). Application layer DDoS attacks include HTTP and DNS attacks.

- *DNS flood attack* attempts to exhaust the targeted server resources with a flood of UDP requests, generated by scripts executed on multiple compromised botnets.

- *DNS amplification attack* does not require a lot of resources from the attacker. It uses spoofing, reflection, and amplification, which means that a small DNS request can be largely amplified in order to result in a much larger response in bytes. As a result, it can dramatically alter the victim's server performance or shut it down completely.

- *HTTP Flood Attack* aims to overload a specific website or web server. It is complex and difficult to detect because an attacker can use random dictionary lookups for news, gov, etc., which look like legitimate requests. These requests use up server resources, causing the website to crash. These requests can also be sent by botnets, which increases the power of the attack.

Network layer attack: It targets layers 3 and 4 of the OSI model. This attack attempts to slow down or disrupt a program, service, computer, or network, or to fulfill its power and make it unavailable to legitimate users. This type of

DDoS attack is based on layer 3 and 4 protocol stacks. It uses the weaknesses of these protocols to consume server resources, or those of intermediate communication equipment, such as firewalls and load balancers. This type of attack is measured by gigabits per second (Gbps) or packets per second (Pps) because typically the traffic is very high.

In this category, we can include types of attacks such as Internet Control Message Protocol (ICMP) flood, User Datagram Protocol (UDP) flood, Synchronize Sequence Number (SYN) flood, and Ping of Death.

- *UDP flood attack* allows attacker to flood random ports on victim's server with UDP packets. These packets cause the server to constantly check the application that is listening on that port and return an ICMP 'Destination Unreachable' packet because no application is found. As more and more UDP packets are received and responded, the system is overloaded and not responding to legitimate users.

- *TCP SYN flood attack* exploits the weaknesses of the normal Transmission Control Protocol (TCP) three-way handshake, which is the communication between clients and the server. The attacker sends multiple SYN requests to the target without responding to the SYN-ACK responses from the server. The server under attack will wait for acknowledgement of each request, which consumes more and more server resources until no new connections can be established, and ultimately results in denial of service.

- *ICMP flooding attack* allows the attacker to bring down the targeted server by flooding it with spoofed ICMP echo requests sent from a huge amount of source IP addresses. As a result, the server's resources are exhausted and therefore unavailable to process legitimate requests. Additionally, this attack uses significant system network bandwidth. The methods used to perform this type of attack, including the use of custom tools or code, such as hping and scapy.

According to the objective of this thesis, we will detail the two types of DDoS attacks: SYN flood attack and UDP flood attack.

1. **SYN flood attack**

   SYN flood attack is a denial-of-service method which exploits the design of TCP three-way handshake for making connections. This attack aims to exhaust a server's allocated state for a listening server application's pending connections to prevent legitimate connections from being established with the server application.

   Every client-server conversation begins with a standardized three-way handshake (Figure.1.9). In the first step, the client requests a connection by sending SYN (synchronize) message to the server, then server responds to the client request with SYN-ACK (synchronize-acknowledge) message, and finally client acknowledges by sending an ACK (acknowledge) message back to the server, and the connection is established.

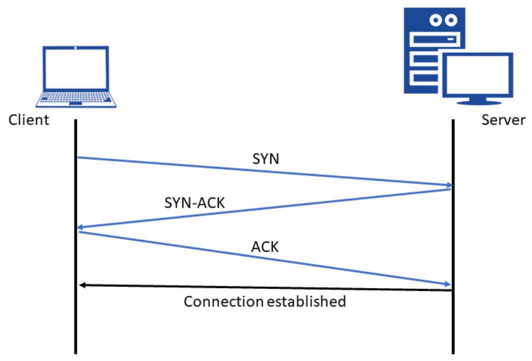


Figure 1.9 – TCP three-way handshake process

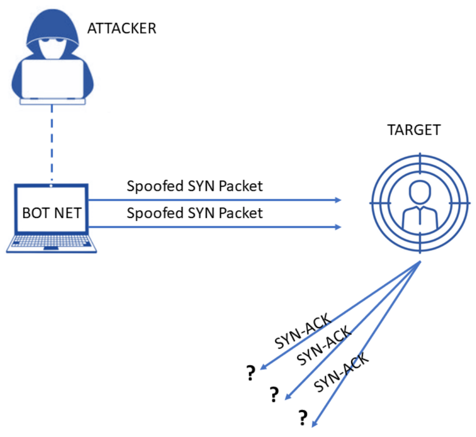


Figure 1.10 – SYN flood attack diagram

SYN flood attack, as shown in Figure.1.10, exploits the TCP implementation by sending an overwhelming number of SYN requests and intentionally never responds to the server's SYN-ACK messages. The server reserves memory space for each requested TCP connection and waits for acknowledgment from the client. By sending SYN segments quickly to a server, the attacker causes the server connection resources to be exhausted, and incoming legitimate connection requests are then denied until the attacker's connections time out.

2. **UDP flood attack**

The UDP flood attack is based on the UPD protocol. UDP is unreliable and connection-less Transport Layer protocol. Since UDP doesn't need to establish connection prior to data transfer, it runs with lower overhead and latency. Therefore, UPD is ideal for real-time services such as voice or video communication, live conferences, etc. UPD packet has a fixed header of 8

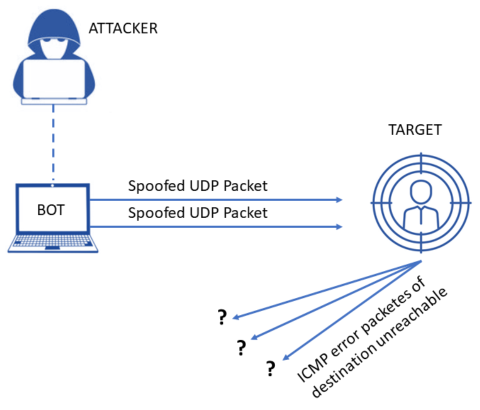


Figure 1.11 – UDP flood attack diagram

bytes unlike TCP header which may vary from 20 bytes to 60 bytes. The header has four fields, each of which is 16 bits (2 bytes), including source port, destination port, checksum and length. Source port number field used to identify port number of the sender (if not used, this field is equal to zero). Destination port number identifies the port of the receiver and it is required. Length field specifies the length (in bytes) of UDP including header and the data. The checksum field may be used for error-checking of the header and data. This field is optional in IPv4, and mandatory in IPv6. Unlike TCP, Checksum calculation is not mandatory in UDP. No Error control or flow control is provided by UDP. Therefore, UDP depends on IP and ICMP for error reporting. As each new UDP packet is received by the server, it goes through steps in order to process the request. When the server receives a UDP packet at a particular port, it checks to see if any programs are listening for requests at that port. If no programs are receiving packets at the specified port, the server responds with an ICMP packet to inform the sender that the destination was unreachable. UDP flood attack exploits these steps to overload the server's ability to process and respond. During the UDP attack, the attacker will spoof the source IP address of the UDP packet, which prevents the attacker's real location from being shown and potentially saturated with response packets from the targeted server, as shown in Figure.1.11. This causes the server to consume its resources to control programs listening in each port and generate a large number of ICMP (Unreachable Destination) packets to respond to UDP flood attacks.

1.3 Security challenges in SDN-based cloud environment

Integrating SDN with cloud computing has many advantages over traditional network infrastructure, such as improved network flexibility, programmability, and scalability. However, this integration poses several security problems. In this section, we first present the crucial security challenges defined separately in cloud computing and in SDN. Then, we discuss the new security issues introduced over the integration of SDN and cloud computing.

1.3.1 Cloud computing security challenges

With the high complexity of network connections in large scale, cloud computing becomes more vulnerable to both traditional and new security issues. The Cloud Security Alliance (CSA) is a not-for-profit organization with a mission to defining and raising awareness of best practices for offering security assurance within cloud computing environment [52]. Recently, CSA realized a survey of industry experts to assemble professional views on the top security challenges in cloud computing, in order to identify the greatest threats [53]. Among the critical security issues identified by CSA, we list the following:

- **Insecure interfaces and APIs:** Cloud computing providers deliver services to their clients through software user interfaces (UIs) or application programming interfaces (APIs). Provisioning and management of cloud services are all made with these interfaces. Therefor, the security and availability of cloud services are dependent upon the security level of these APIs. The UI, API functions and web applications share a number of vulnerabilities, which may cause various attacks related to the confidentiality, availability, integrity, and accountability of cloud services. Thus, any interfaces that will connect to cloud infrastructure must be designed with strong authentication methods (e.g., SSL/TLS), proper access controls, and encryption methods.

- **System Vulnerabilities:** No system is 100% secure: every system has vulnerabilities, which can be exploited to negatively impact confidentiality, integrity, and availability of provided services. Since virtualization is a key technology in cloud infrastructure, any vulnerability can put the security of the system and all services at significant risk. For instance, any fault and vulnerability within the hypervisor may be harnessed to launch virtual machine (VM) attacks or monitor the shared resources of the present VMs.

- **Service interruption:** Since everything in the cloud is defined as service provided on-demand, the availability is the crucial security requirement in cloud computing. Deny of service (DoS) and distributed DoS (DDoS) attacks are the main threats to interrupt availability of cloud services. DDoS attacks may happen when an attacker forces a targeted cloud service to use excessive amounts of finite system resources like network bandwidth, memory, CPU, or disk space, which render services and computing resources unavailable.

- **Advanced persistent threats:** Advanced Persistent Threat (APT) is a parasitic form of the cyber attack that infiltrates systems or networks and remains there for an extended period of time without being detected. The intention of an APT attack is usually the monitoring of the network activity and stealing valuable data rather than causing harm to the network or company. Although

APT attacks can be difficult to detect and eliminate, some can be stopped with system monitoring, proactive security measures, and awareness programs.

1.3.2 SDN security challenges

Although SDN characteristics help in protecting cloud environment against various threats, SDN itself suffers from both the present security attacks and new issues. Due to the centralized control and the network programmability of SDN, new security challenges have been introduced across SDN layers [54], [26], [55], [56].

- **Application layer:** Unauthenticated and unauthorized applications are one of the serious security breaches in SDN. They may access and modify network data or reprogram the SDN components. Consequently, authorization and authentication of these applications are needed to defend network resources against malicious activities.

- **Control layer:** Since SDN controller is the brain of SDN architecture, the majority of SDN security issues are related to the control plane vulnerabilities. Due to the separation of the data plane and the control plane within the SDN framework, the controller itself may become a target for various threats like flooding and DDOS attacks. An attacker can initiate a resource consumption attack (e.g., SYN flood attacks) on the controller to make it unavailable in response to the switch requests.

- **Data layer:** The lack of SSL/TLS adoption within controller-switch communication may cause access of unauthorized controller and so insertion of fraud flow rules in SDN OpenFlow switches. These latter suffer also from saturation attacks (i.e., flow-table overloading attacks) due to the limited storage capacity of the switch flow tables.

- **Insecure interfaces and APIs:** Since all communications between the application, control and data layers, or even the communication between multiple controllers passe through Application Programming Interfaces (APIs) (i.e., Northbound, Southbound, and East and West Interfaces), it is primordial to secure them. The different Northbound interfaces share a number of vulnerabilities, which may cause various attacks related to the controller availability, and network elements processing. As OpenFlow protocol is the standard southbound interface of SDN, it suffers from the lack of TLS adoption by major vendors which could lead to malicious rule insertion and rule modification. In a multi-controller environment, controller's communication passes through the East/West interfaces. Most of the time these controllers don't share a common secure channel (i.e., controllers from different vendors) between them, which could lead to sniff important messages and expose sensitive information.

1.3.3 Security issues of SDN-based cloud environment

Although SDN-enabled cloud computing has great benefits in comparison to the traditional cloud networking, it poses several security issues [16], [26], [5], [11] that we have discussed bellow (presented briefly in Table 1.1).

- **Availability:** Since everything in cloud is defined as service, availability is a crucial security requirement which is directly related to the availability of SDN technology. There are two major availability issues rely on SDN architecture: a) In the network expansion, the SDN centralized controller may become a target for various threats like DDOS attacks) The flow tables of SDN switches suffer from saturation attacks due to their limited storage capacity.

- **Scalability:** Scalability of SDN-enabled cloud relies on the scalability of the networking infrastructure which is controlled by the SDN technology. In the current growth networks, the SDN centralized controller may easily become a bottleneck which may disrupt cloud evolution.

- **Authorization and authentication:** In SDN-enabled cloud architecture most of the network services are presented as third-party applications, which have access to network elements and can also handle network functions. Consequently, authorization and authentication of these applications are needed to defend network resources against malicious activities.

- **Fraudulent flow rules:** Due to the isolation of decision-making functions from the SDN switches, these latter can not identify the legitimate flow rules from fraud flow rules. Hence, an attacker may easily insert fraudulent flow rules within the switches while exploiting vulnerabilities of the southbound interfaces.

- **DDoS:** DDoS attack is one of the most challenging security concerns for SDN. Due to the separation of control decisions from the forwarding plane, each new flow is transmitted to the controller. Therefore, DDoS attackers can easily exhaust the controller resources while sending a large number of new flows. Once the controller is deactivated, all network services are also deactivated.

Although, security is among the major concern in SDN-enabled Cloud computing, there are a limited number of works [16], [26], [5], [11], [57], [58] which analyzed and examined the security challenges of the SDN-based cloud. In [11], Yan et al. discussed the new DDoS attacks tendency and features in cloud computing and supplied a comprehensive study of SDN-based defense mechanisms against DDoS attacks. This study gives us a clear view of how to make the SDN characteristics useful to protect cloud systems from DDoS attacks and how to prevent SDN itself from becoming a sufferer of DDoS attacks. Son et al. [5] presented a taxonomy of the usage of SDN in cloud computing in various aspects including energy efficiency, performance, virtualization, and security enhancement. In [16], the authors discussed the SDN feasibility in the cloud environment and represent the flow table-space of a switch by using a queuing theory based mathematical model. They presented a novel flow table sharing approach to defense the SDN-based cloud against flow table overloading attacks. Farahmandian et al. [26] presented major security challenges in cloud, SDN, and NFV and suggest solutions using virtualization technology. The paper discussed the need for a software-defined security technology for managing an integrated infrastructure platform where cloud, SDN, and NFV all play their integral parts. In [58], Khedkar et al. presented a survey on SDN enabled technologies

Table 1.1 – Security challenges of SDN-based cloud environment

Security challenges	**Description**
Scalability	Scalability of SDN-enabled cloud relies on the scalability of the networking infrastructure which is controlled by SDN technology. In the current growth networks, the SDN centralized controller may easily become a bottleneck which may disrupt cloud evolution.
Availability	Since everything in cloud is defined as service, availability is a crucial security requirement which is directly related to the availability of SDN infrastructure.
Authorization and authentication	In the application plane, unauthenticated and unauthorized applications pose a great challenge for SDN.
Fraudulent flow rules	Exploiting the vulnerabilities of the southbound interfaces, the attacker can easily insert fraudulent flow rules within the switches.
Flow table overloading	The flow tables of OpenFlow switches suffer from saturation attacks due to their limited storage capacity.
DDoS	Due to the isolation of decision from the data plane, each new flow is forwarded to the controller. Therefore, DDoS attackers may exhaust easly the controller resources while sending a great number of new flows.

over cloud, IoT and data center networks (DCNs). They also offered insight into future research efforts for SDN enabled technologies. In [57], a study conducted to realize a review of SDN security issues, including network virtualization and cloud infrastructure, to conduct a systematical survey of SDN controllers, and to focus on SDN functionality supporting security.

1.4 Impact of DDoS attacks in SDN-based cloud environment

With the adoption of cloud services, the rate of DDoS attacks against cloud infrastructure increases since the traditional DDoS attacks defense techniques are unable to protect the large-scale network of cloud data center. Some significant characteristics of SDN approach help to defend the DDoS attacks in the cloud computing environment. However, SDN itself may be targeted by the attackers, which raise the risk of DDoS attacks in the SDN-based cloud. In this section, we first discuss how the cloud characteristics make it more vulnerable to DDOS attacks. Then, we present the possible DDoS attacks on SDN. And finally, we discuss some SDN features which improve the DDOS attack mitigation in Cloud.

1.4.1 DDoS impact in cloud computing

A recent survey by the Cloud Security Alliance (CSA) shows that DDoS attacks are critical threats to cloud security [59], [41]. In addition, a set of studies [15], [60], [11] shows how the essential characteristics of cloud computing can be the reason for increasing the rate of DDoS attacks in a cloud environment, as we can see below:

- `On-Demand Self-Service:` The on-demand Self-Service ability allows clients to independently get cloud services (e.g., computing, storage, network, etc.). This feature can be easily exploited to build a powerful botnet and initiate DDoS attacks in a short time. Therefore, the distributed DoS attacks increased as the large-scale botnets increased in the cloud environment.

- `Broad Network Access:` Broad Network Access feature allows customers to access cloud services through different mobile devices, such as smart-phones and tablets. The lack of security on the majority of these devices can be used to launch DDoS attacks in cloud Infrastructure.

- `Resource Pooling:` Virtualization technology is a key development of cloud computing. Therefore, any virtualization vulnerability can put the security of the cloud system and all services at significant risk. The multi-tenant model is utilized in the cloud to pool physical and virtual resources among multiple tenants. The vulnerabilities of multi-tenant and virtualization technologies can be exploited to easily launch DDoS attacks and make the system more vulnerable to DDoS threats.

- `Rapid Elasticity and Measured Service:` The cloud's pricing model enables clients to pay for their use of the cloud's services. In combination with rapid elasticity, the pricing model can be used to financially affect a customer by generating fraud invoices. Such threat can also be planned to perform an Economic Denial of Sustainability attack (EDoS), which is a new form of DDoS attacks.

1.4.2 How SDN's features may enhance the DDOS defense in Cloud?

In traditional networks, it was difficult to implement, experiment, and deploy new ideas on large-scale networks such as cloud environment. However, the separation of the control plane from the data plane in SDN enables experimenters to perform easily large-scale attack and defense experiments in a real environment [11]. This separation provides a programmable network in which network devices can operate and manage network traffic dynamically [17]. Hence, the programmability of SDN allows us to flexibly implement intelligent defense algorithms against DDoS attacks in cloud environment. A centralized control feature of SDN gives a network-wide knowledge, which helps to build a relevant security policy for the network. Characteristics like centralized control and programmability allow SDN to defend cloud computing against DDoS attacks [19]. In SDN, the network traffic can be analyzed innovatively using intelligent mechanisms and various types of software tools [18]. Hence, SDN can greatly enhance the DDoS detection and mitigation ability using the software-based traffic analysis.

1.4.3 DDoS impact in SDN

SDN features present a great promise in terms of defending against threats in large networking environments. However, SDN itself have many security challenges including, flow-table overloading, Controller-aimed Distributed Denial of Service (DDoS) attacks, unauthorized access, Fraudulent flow rules, malicious applications, data leakage, etc [18], [8], [16], [20]. DDoS attacks which have been completely covered by the security community, today pose potential new menace to the availability and scalability of the SDN network management. The SDN architecture is divided into three planes including, application plane, control plane, and data plane. All these planes and Application Programming Interfaces (APIs) can be targeted by the attackers to launch DDoS attacks.

- **Control plane:** The controller is the brain of the SDN architecture which provides central control over the network. Hence, it could be seen as a single point of failure if it is made unreachable by a Distributed Denial of Service (DDoS) attack. For example, an attacker can make the controller unavailable while producing a series of new TCP SYN requests, from distributed bot clients, which involves the controller in a series of useless processes. In this case, the controller will be saturated and so unable to process legitimate requests. In the control plane, DDoS attacks can target controller services, northbound interface, southbound interface, eastbound interface, and westbound interface.

- **Data plane:** The state-full implementations and functionalities in SDN like connection tracking (conntrack) [61] in Open vSwitch, OpenState [62], registers and counters in P4 [4], all require storing the state of flow in the data plane. Therefore, forwarding elements are again vulnerable to the saturation attack.

- **Application plane:** In the application plane, unauthenticated and unauthorized applications pose a great challenge for SDN. An attacker can launch an unauthenticated and unauthorized application with malicious programs running on the network devices and so the attacker can easily gain control of the network. Furthermore, the problem of isolation of applications and resources is not well solved in SDN; as a result, a DDoS attack on one application can affect other applications as well.

1.5 The current DDoS detection and mitigation mechanisms in SDN-based Cloud environment

SDN-enabled cloud presents many advantages in comparison to the traditional networking infrastructure, such as improved network scalability and flexibility, improved network programmability, etc... However, the recent assessments [16], [18], [19], [20] and our analysis in sections 1.3 and 1.4 show the introduction of new security issues and particularly new trends of DDoS attacks over the integration of SDN and cloud computing technologies.

To the best of our knowledge, there are limited research works which address the potential challenges to mitigate DDoS attacks in the enterprise networking that adopts both cloud computing and SDN technologies. Among which we present and analyze the following works, with a brief description in Table 1.2.

Bhushan et .al [16] presented the SDN principle and SDN-enabled cloud. They discussed the DDoS impact in the SDN-based cloud and the existing solutions to protect cloud environment from DDoS attacks using SDN. They presented a new approach to defense the SDN-based cloud against flow table overloading attacks. The approach utilizes the unused flow table-spaces of other switches to resist the attack at a particular switch. To increase the availability of flow tables in SDN, the proposed mechanism removes less utilized flow rules and flow rules belonging to the attack traffic. They maintain two databases for the operation of there approach: Flow Table Status and Black List. Flow Table Status records the current status of flow tables of all the switches in the network, i.e. the number of entries occupied in each flow table. The Black List database lists the IP addresses of attack sources. The objective of this approach is to enhance the resistance of the SDN against DDoS attacks while increasing the time to overload all the SDN switches (i.e., the holding time) at least up to the reaction time of DDoS defense systems. The experimentation showed that the holding time of SDN has been significantly improved when the proposed approach is applied and the communication between controller and switch have been reduced during the attack. However, if there are a large number of new TCP connection attempts from different and new IP sources the switch will ask the controller for new flow rule for each connection. Hence, the SDN centralized controller may easily become a bottleneck which may disrupt the network services.

Wang et .al [18] studied the impact of SDN-enabled cloud on DDoS attacks defense. Based on their analysis, they remarked that if the DDoS attacks mitigation solution in SDN is designed correctly, SDN will benefit the DDoS attacks protection in cloud computing environment. Therefore, they proposed a DDoS attacks defense architecture (DaMask), which contains two modules: an anomaly-based attack detection module DaMask-D, and an attack mitigation module DaMask-M. For DaMask-D module, they developed an attack detection system which is built on probabilistic inference graphical model. The proposed detection system advances with two capabilities; an automatic feature selection to build an effective graph model and an efficient model update to address the dataset shift problem. DaMask-M is a flexible control structure which allows quick attack reaction. The authors evaluate the DaMask architecture under a private cloud and a public cloud (i.e., the Amazon Web Service (AWS)). The evaluation indicated that the proposed at-

Table 1.2 – The current DDoS detection and mitigation mechanisms in SDN-based Cloud

Related work	**Description**
Distributed denial of service (DDoS) attack mitigation in software defined network (SDN)-based cloud computing environment	Bhushan et .al discussed the SDN feasibility in the cloud environment and represent the flow table-space of a switch by using a mathematical model based on queuing theory concepts. They presented a new approach to defense the SDN-based cloud against flow table overloading attacks. This approach utilizes the unused flow table-spaces of other switches to resist the attack at a particular switch.
DDoS attack protection in the era of cloud computing and software-defined networking.	Wang et .al studied the impact of SDN-enabled cloud on DDoS attacks defense. They remarked that if the DDoS attacks mitigation solution in SDN is designed correctly, SDN will benefit the DDoS attacks protection in cloud computing environment. Therefore, they proposed a DDoS attacks defense architecture 'DaMask', which contains two modules: an anomaly-based attack detection module DaMask-D, and an attack mitigation module DaMask-M.
DDoS attack detection and mitigation using sdn: methods, practices, and solutions.	Bawany et .al made a survey of the current SDN-based DDoS attack detection and mitigation strategies. Motivated by the observed requirements, they proposed an SDN-based proactive DDoS Defense Framework (ProDefense) for smart city data center. The approach enables the implementation of different application-security requirements and also has distributed controllers which increase the reliability and scalability of the solution.
Effective software-defined networking controller scheduling method to mitigate DDoS attacks.	Yan et .al developed a multi-queue SDN controller scheduling method to mitigate DDoS attacks in SDN. The proposed method defend the normal switches during a DDoS attack and prevent the SDN network from being unavailable by programming the flow request processing through different switches. This approach uses different time slicing strategies depending on the DDoS attack intensity.

tack detection algorithm gives a cheaper computation cost of attack detection and a constant communication overhead as long as the link status of the network is stable. Additionally, the evaluation validated the DaMask ability to adapt to the network topology change caused by virtual machine migrations. Moreover, it exhibited that the local model update mitigates the impact of the dataset shift problems and therefore, improves the detection accuracy.

Bawany et .al [19] made a comprehensive and extensive survey of the SDN-based DDoS attack detection strategies. Motivated by the key requirements of an effective DDoS attack prevention mechanism, they proposed an SDN-based proactive DDoS Defense Framework (ProDefense) for smart city data center. Since the smart city applications have different security requirements (i.e., catastrophic, critical, moderate), ProDefense framework uses a customized detection filter to meet the security requirements of various application-specific. The distributed control layer ability enhances the reliability and scalability of the proposed solution to support the evolution of smart city data centers. Further, the article discussed the open research challenges, future research directions, and recommendations related to SDN-based security solutions.

Yan et .al [20] developed a multi-queue SDN controller scheduling algorithm 'MultiSlot' to mitigate DDoS attacks in SDN. The proposed method defends the normal switches during a DDoS attack and prevents the SDN network from being unavailable by programming the flow request processing through different switches. This approach uses different time slicing strategies depending on the DDoS attack intensity. The simulation results showed that the proposed method has better performance than existing schemes in terms of protecting the internal switches that are affected by the DDoS attacks. The simulation results showed that the proposed method has better performance than the existing methods in terms of protecting the internal switches that are affected by the DDoS attacks.

1.5.1 Discussion

All the above-mentioned DDoS mitigation mechanisms are designed at the high-level SDN application plane with the involvement of the SDN controller in each operation to detect and mitigate DDoS attacks. Therefore, the communication path between the data and control planes rapidly becomes a bottleneck, which may impact the network performance and restricts its scalability and reactivity. In this side, the SDN community has taken into consideration the necessity to conduct specific functions directly inside the SDN Switch [63], [64], [65]. SDN data plane based specific functions can be used to minimize the switch-controller communication and optimize the scalability and responsiveness of the defensive mechanisms.

Frameworks, Compilers, and Programming Languages [21], [66], [67], [22], [23], [24] have been developed to take advantages of the SDN data plane with a way to perform dynamically specific operations (e.g., monitoring, detection, reaction, etc.) inside the switch.

Programming Protocol independent Packet Processors (P4) [4] is a domain-specific programming language designed for describing how packets are processed by the data plane of a programmable forwarding element, such as a hardware or software switch, router, network interface card, or network appliance. Interested P4-based security solutions have been developed. Voros et .al [68] discussed the primary security middle-ware programmed and configured in P4. They proposed a stateful firewall for mitigating network flooding attacks using P4 language. Afek et .al [69] suggested a defense system against network spoofing attacks while performing selected anti-spoofing techniques in OpenFlow 1.5 and P4 (i.e., match and action rules). They also developed dynamic algorithms for automatic distribution of sophisticated rules on network switches.

As another programming language and compiler for data plane, SNAP was presented by [23] which allows stateful network-wide abstractions for packet processing. It defines some state variables and arrays to install rules and to maintain the state information of the flows. The authors in [24], have proposed Domino compiler for packets processing in Data Plane. The packet transactions are compiled by the Domino and executed on Banzai machine to allow high-level Programming for Line-Rate Switches.

1.6 conclusion

In this chapter, we attempt to give a general overview of cloud computing, its architecture, service models, deployment models, and characteristics. Then we presented SDN architecture and we discussed their features which make it an appropriate technology for cloud system. For example, centralized control ability allows a global view of the network, and provides centralized network management and provisioning in the cloud computing environments. Afterward, we analyzed the new security issues introduced over the integration of SDN and cloud computing technologies. Subsequently, we examined how the cloud and SDN characteristics make them more vulnerable to DDOS attacks. At the end of this chapter, we reviewed and analyzed the recent proposed research approaches and we presented an interesting state of the art on DDoS detection and mitigation in the SDN-based cloud environment.

Based on our analysis, we defined the challenges and opportunities raised by these new technologies. We claim that with a careful design of SDN-based DDoS detection and mitigation, SDN will benefit the DDoS attack protection in cloud computing. In the next chapter, we will present the proposed mechanism 'SDN-based SYN flooding defense in cloud' to prevent and mitigate SYN flooding attacks in the cloud environment.

Chapter 2

SDN-based SYN Flood Defense in Cloud

2.1 Introduction

Cloud Computing has been introduced as the next generation architecture of IT companies and it gives great capabilities that ensure improved productivity with minimal costs. Cloud Computing provides an enhanced level of flexibility and scalability in comparison to the traditional IT systems. The openness of the cloud environment makes it very complex and unpredictable, which poses many security challenges and attract the attention in various research works.

As discussed in Subsection.1.4.1 due to their characteristics, the cloud system becomes more vulnerable to various security threats and particularly to DDoS attacks. The outstanding question to be answered is how to defend against such attacks in cloud environment. Many security solutions have been used to ensure cloud security while analyzing the normal from the abnormal user/network behavior so as to detect the cloud users which are untrustworthy.

Network Intrusion Detection and Prevention Systems (NIDPS) [70], [71], [72] have been proposed as a suitable security solution for attack detection and mitigation in the network. Although the usage of the traditional NIDPS exhibits the network to significant problems due to the high consumption of resources and the delay in response time, as well as they are remained limited into the management and security of scalable networks [73], [74].

The emergence of the SDN paradigm offers significant benefits for managing and securing large and scalable networks like cloud. SDN capabilities including network-visibility, centralized control, programmability, software-based traffic analysis facilitate the implementation, configuration, and control of network security functions, such as firewall, intrusion detection system (IDS), intrusion prevention system (IPS), etc.

In this sense, many research works have proposed SDN-based approaches to improve the network security and most of them are implemented at the control plane level [75], [76], [77]. As a result, the communication path between data and control planes quickly becomes a bottleneck, affecting network performance and limiting its scalability and responsiveness. To deal with these limitations, a few works [78], [79] have proposed security extensions that fundamentally apply flow management at the switch level. However, implementing these extensions requires unsupported changes in OpenFlow switches.

Recently, a new programming network language named Programming Protocol-independent Packet Processors (P4) [4] has been designed, it describes how packets are handled by the data plane. Many research experiences indicate that the implementation of novel applications which operate with new header fields in OpenFlow environment required complex design and several lines to add and modify compared to the programming in P4-enabled switches [7].

In this sense, we propose the design of an active defensive mechanism which enables the data plane to detect and mitigate DDOS attacks and particularly SYN flooding attacks in cloud environment. Our proposal exploits the capacities of SDN technology and takes advantage of the switch programmability using P4 language.

The main objectives of this contribution can be summarized as follows:

- Activate the data plane with customized stateful databases which are useful for the security applications to monitor traffic and flow rate anomalies.
- Enable the data plane to prevent the overwhelming traffic attacks (e.g. DDoS attack) at an early stage using the traffic anomaly detection algorithm (CUSUM).
- Add intelligence to the data plane to mitigate SYN flood packets using SYN cookie methods, then deploy adaptive countermeasures.

The remainder of this chapter is structured as follows: In Section 2.2, we discuss the proposed SDN-based security solutions, their limitations and the Stateful SDN Data Plane Applications proposed to deal with these limitations. The Section 2.3 presents and describes the methods and techniques that will be used in our proposed mechanism. We present our approach 'SDN-based dynamic defense mechanism against SYN Flooding attacks' in Section 2.4. Finally, we give our conclusion in Section 2.6.

2.2 Related Work

2.2.1 SDN-based security solutions

By decoupling the data and control planes of traditional networks, SDN appears as an emerging, agile and flexible technology that gives hope to deal with the static network architecture and to facilitate the network management [80]. For example, Yoon et al. [81] suggested enabling security functions (such as Firewall, NIDS, etc.) with SDN so as to demonstrate the applicability of the SDN-based security applications. Their aim is to promote researchers and practitioners to develop robust security solutions in SDN. Considering the IDPS, in [74],the authors proposed SDNIPS as SDN-based Intrusion Prevention System in the cloud virtual networking environment. Their solution uses the Snort IDS and the SDN architecture to offer a dynamic mechanism for security assessment in cloud environment. Seunghyeon et al. developed a scalable framework, called Athena, as an anomaly detection application that uses SDN functionality to explicitly support Machine Learning-based network anomaly detection. Athena's API offers a well structured development environment to implement new anomaly detection services across SDN infrastructure. However, these works develop specific high-level SDN applications to provide better security solutions. Therefore, the communication channel between the data and control planes rapidly becomes a bottleneck, which can degrade the network performance and limits its scalability. To address these issues, the SDN community introduces the new programmable data plane (see Subsection. 1.2.3) that supports more advanced features. According to the literature, there are a restricted number of works that basically discuss flow management at SDN switch level and describe its processing. For instance, OpenFlow Extension Framework (OFX) [79] activated practical SDN security applications within unmodified OpenFlow data plane. OFX enables applications to dynamically charge software modules into the existing OpenFlow switches where application process (e.g., monitoring, detection) can perform closer to the data plane. Shin et al. [82] proposed Avant-guard as a detection and prevention solution against the TCP SYN flooding attack. Avant-guard extends the data plane with the connection migration module which proxying the incoming TCP SYN packets and prevent the control plane from saturation attack. LineSwitch [83] is an improved Avant-guard solution. LineSwitch addressed the vulnerabilities and limitations of Avant-guard by proxying a minimum number of TCP SYN requests. The implementation of [82, 83] solutions requires complex design and several lines to add and modify to extend and customize the data-plane. Moreover, They are based on fixed-function switch which offers a complex design to enable variations in data plane pipelines. Unlike the traditional fixed-function Openflow-enabled switch, P4-programmable switch is new concept to make the SDN network more flexible, dynamic, programmable, and scalable. In this research work, we choose to work with the P4-programmable switch to take full advantage of the SDN technology.

2.2.2 Stateful SDN data plane applications

As mentioned above, the SDN community has taken into consideration the necessity to perform specific functions directly inside the SDN Switch. To the best of our knowledge, there are few publications that introduce and discuss emerging stateful SDN data plane proposals. In [63] ,the authors surveyed the recent proposed solutions for the stateful SDN data plane [23], [67], [22], [21] where their hope is to deal with the limitations of OpenFlow and ensure a high-level abstraction for the data plane with way to configure dynamically the stateful operations inside the switch. In addition, they discussed the key potential vulnerabilities related to these stateful proposals that can occur due to the limited flow state memory allocation, non-effectiveness of the authentication mechanisms and policies or lack of a central state management. Finally, they have given a specific scope for practical attacks examples that can take benefits from these vulnerabilities. OpenState [67], FAST(Flow-level State Transitions) [66] and SDPA [22] are stateful data plane abstraction. These platforms are designed to handle the flow states inside the switch using finite state machines as an extension of OpenFlow protocol. These programmable SDN switches have the capability to locally use stateful rules in order to improve the network performance and the responsiveness to the real-time network applications.

2.3 Methods and Techniques

2.3.1 Anomaly detection methods

Due to the fluency and flexibility of the anomaly detection algorithm implementation, many researchers have studied and implemented these methods to detect DDoS attacks in networking systems.

DDoS flood attack is an amplification of normal network packets, making it difficult to detect. Statistical techniques, as type of Anomaly detection methods, are the most powerful methods to detect DDoS flooding attack because they don't assume any previous knowledge about the network behavior.

Many anomaly detection algorithms have been developed, for example Cumulative Sum (CUSUM) which is the main anomaly detection algorithm that is widely used for detecting DDoS attacks [84, 85, 86].

A. Cumulative Sum (CUSUM) algorithm

CUSUM algorithm is based on change point detection theory [6]. Its principle is to trigger an alarm when the accumulated volume of measurements during a certain time exceeds some total volume threshold (see Figure. 2.1). The CUSUM can be used as a perfect algorithm for DDoS attacks detection.

There are a number of variations of CUSUM technique. For example, CUSUM has been proposed by Wang et al. [87], as a statistical technique, to detect SYN flood attack. The SYN flooding attack is detected by examining the number of inactive RSTs flags by measuring the difference between the total number of SYN and FIN flags. In a normal connection, if the socket is closed by sending a packet, a RST packet is sent which is interpreted as an active RST. The system computes the possibility of a SYN flood attack by measuring the number of inactive RSTs and comparing it to a threshold value. An attack is announced when an unexpected change has occurred in the inactive RST's.

Figure. 2.1 illustrates the CUSUM behaviors. Let X_n a number of packets collected during an elapsed time Δ_n. $\overline{X}$ is the mean value of X, where $X = \{X_n, n = 0, 1, 2, 3, ...\}$. Let the $Z_n = X_n - \alpha$, where α is the peak value of normal traffic. So, $\overline{Z}$, the mean value of $Z = \{Z_n, n = 0, 1, 2, 3, ...\}$, Z must be negative during normal operation.

When DDoS flood attack occurs, Z_n will become large positive very quickly, and $Z_k \geq Z + h$, where k is the starting time and h is the abnormal traffic network threshold. We accumulate Z_k with the formula:

$$y_n = (y_{n-1} + Z_k)^+, y_0 = 0 \tag{2.1}$$

Where N is the attack threshold. If the accumulative value $y_n > N$ at some time point after the starting time, we can conclude that there is a DoS or DDoS attack.

In recent years, various approaches have been proposed using the CUSUM algorithm for detecting DDoS attacks. For example, in [84], the cumulative sum (CUSUM) and the adaptive threshold algorithms have been implemented to detect the SYN Flooding attack. The authors evaluated these statistical algorithms in

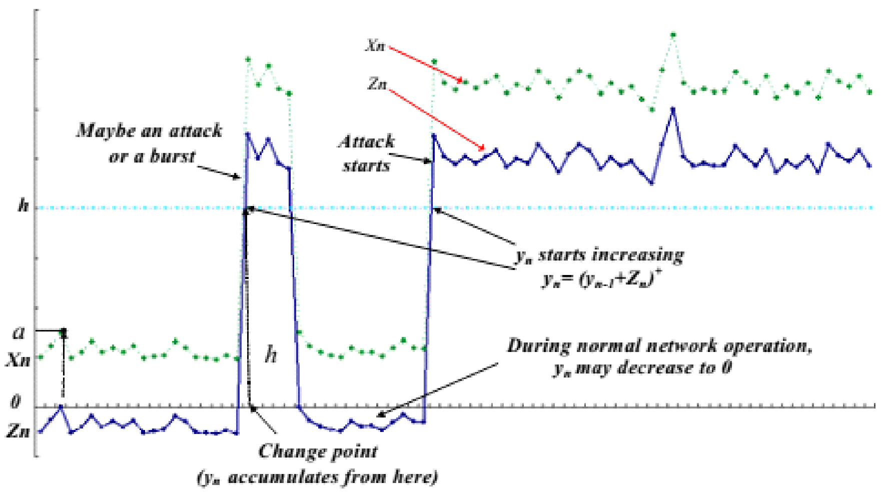


Figure 2.1 – CUSUM's behavior during DDoS flood attack [6]

terms of probability of detection, false alarm rate, and detection delay. They demonstrated the flexibility of these algorithms to tune their parameters so as to achieve better performance. Another work in [6] suggested an IPS called Cumulative-Sum-based Intrusion Prevention System (CSIPS) for DoS/DDoS attacks detection. The proposed system implements the CUSUM algorithm to evaluate the bandwidth consumption attack and detection accuracy.

2.3.2 SYN flooding defense methods

TCP implementation keeps information about every connection between the client and the server. Session information is stored in a structure called 'Transmission Control Block or TCB' of tables. TCP TCB is a set of variables that store information about endpoints (IP address and port), status of the connection, window size and sequence numbers about the packets that are being exchanged, and buffers for sending and receiving data.

SYN flood attack exploits this option of the TCP implementation to exhaust the server resources by overloading it with new connection requests. Receiving SYN flood packets causes the server to allocate a large number of Transmission Control Blocks which can saturate the server and make it unavailable.

Since everything in the cloud is defined as service provided on-demand, the availability is the crucial security requirement in cloud environment. DDoS attacks and particularly SYN flooding attacks are the main threats to interrupt availability of cloud services. Further, SDN suffer from such attack mainly called control plane saturation attack, uses the same SYN packets flood, which now exhausts the bandwidth of the control channel between an SDN controller and a switch , as well as exhausting the switch flow-tables entries space.

The most efficient mitigation methods against spoofed SYN flood attacks are various variations and combinations of the SYN Cookie method.

The SYN cookie [88] is a state-less technique to prevent the memory consumption caused by the half-open SYN attacks (SYN flood attacks). The system (i.e., server or switch) enabling the SYN cookie technique intercepts the SYN request received from a client and sends back a SYN-ACK packet with a pre-generated cookie (i.e., the Initial Sequence Number (ISN)). The cookie or ISN is generated using some details of the initial SYN packet and cryptographic hashing function to make the decoding of the cookie more complicated (see Subsection. 2.3.3). If the mitigating system receives an ACK packet from the client (with pre-generated cookie +1), it checks the validation of the TCP sequence number (i.e., is the ACK-1), as shown in Subsection. 2.3.3.

There are different SYN cookie methods that exist to interact between clients and servers, among which we define; TCP proxy, TCP reset, safe reset, HTTP redirect [7]. In this work, we implement the anti-spoofing methods using P4 and P4Runtime in cloud environment.

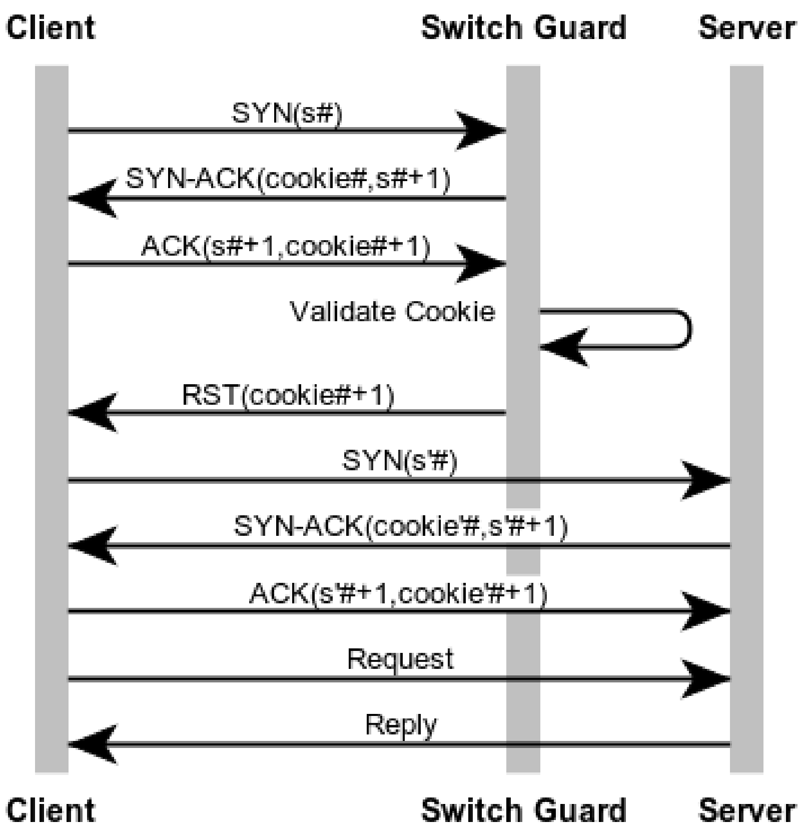


Figure 2.2 – TCP-Reset [7]

1. **TCP-Reset method:** [89], [90] When the client is authenticated (i.e using SYN cookie) the mitigating system classifies it as legitimate (i.e., the system installs a rule by recording the source IP of the connection as legitimate). Then the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client in order to enable him to re-establish the connection directly with the server. The advantage of this method is that is suitable for all TCP connections that attempt to connect when a RST packet is received.

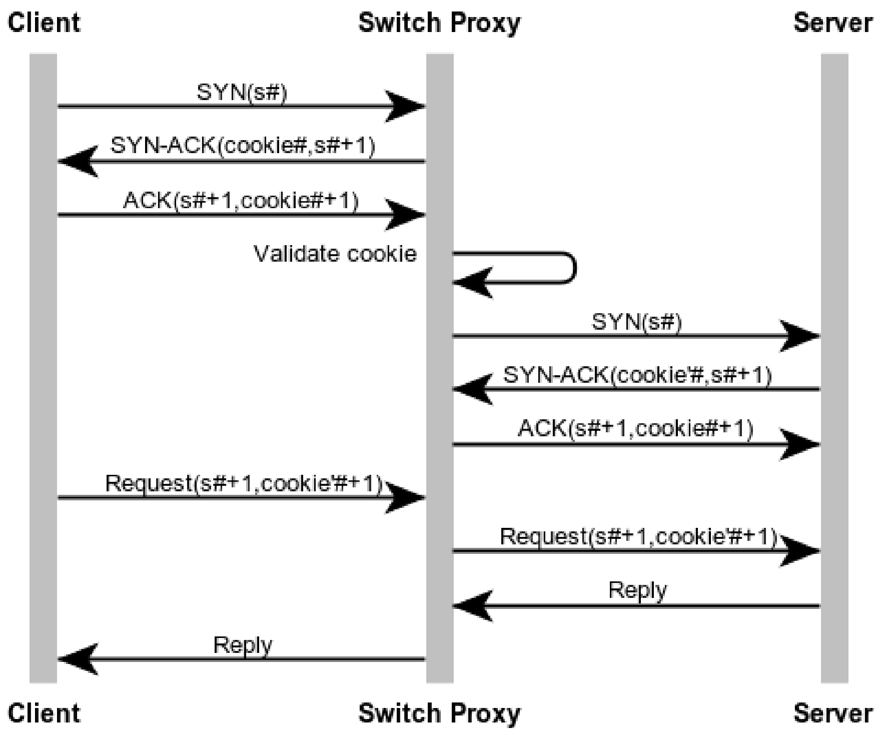


Figure 2.3 – TCP-Proxy [7]

2. **TCP safe Reset:** In this version, the switch responds to the initial SYN message with an SYN-ACK containing a false ACK number. If the client complies with RFC 793 (TCP) [91], it immediately responds to the false ACK number with a RST packet containing the false ACK number as its sequence number (thus being authenticated). Then, 1 second after that, the client initiates a new connection by sending a new SYN request. This technique does not assume any knowledge at the application level and only requires TCP compliance, so it is suitable not only for HTTP protocols, but also for SMTP and others.

3. **HTTP Redirect:** After being authenticated, the mitigating system classifies it as legitimate (i.e., the system installs a rule by registring the source IP of the connection as legitimate), then responds to the HTTP get request with a corresponding HTTP redirect response, and finally closes the authenticated connection. Sending a redirect response with the same original server address forces the client to re-establish the connection. This time it will go directly to the server because of the installed rule.

4. **TCP-Proxy method:** On receipt of a valid ACK packet matching the SYN cookie, the system makes a hand-shake with the server. That is mean that the switch acts as a proxy between the client and server during the entire connection, which may present a significant overhead. For every subsequent packet in each direction, it modifies the sequence numbers so that it is transparent to both sides. The advantage of this method is that it allows us to know the successful and failed TCP connections (server-side).

As a default, SYN cookie technique had to be enabled all the time, since there is no simple way to activate it only when under attack. Inspiring by the Linux kernel method, which automatically enables the SYN cookie only when the SYN queue is full, we enable our system to activate the SYN cookie techniques only when the rate of TCP SYN requests reached the defined adaptive threshold, rather than enabling them constantly.

2.3.3 Pre-generated Cookie

According to [92], the implementation of the SYN cookies must fulfill the following basic requirements:
- Cookies should contain some details of the initial SYN packet and its TCP options.
- Cookies should be unpredictable by attackers. It is recommended to use a cryptographic hashing function in order to make the decoding of the cookie more complicated. For this end, we select the recommended Linux SYN cookies method for generating and validating cookies [93].

Cookie generation:

$$H_1 = hash(K_1, IP_s, IP_d, Port_s, Port_d) \tag{2.2}$$

$$H_2 = hash(K_2, count, IP_s, IP_d, Port_s, Port_d) \tag{2.3}$$

$$ISN_d(cookie) = H_1 + ISN_s + (count \times 2^{24}) + (H_2 + MSS) \mod 2^{24} \tag{2.4}$$

Cookie validation:

$$ISN_d = ACK - 1 \tag{2.5}$$

$$ISN_s = SEQ - 1 \tag{2.6}$$

$$count(cookie) = (ISN_d - H_1 - ISN_s)/2^{24} \tag{2.7}$$

$$MSS(cookie) = (ISN_d - H_1 - ISN_s) \mod 2^{24} - H2 \mod 2^{24} \tag{2.8}$$

As we can see above and in Table. 2.1, we calculate the two hash values H_1 and H_2 (based on TCP options, secret keys k_1, k_2 and count) then we use them with ISN_s and MSS to generate the cookie (ISN_d), as it is shown in (3). For the cookie validation, there are 2 integrity controls (count(cookie) and MSS(cookie)). The first one checks the age of the cookie. The second evaluates whether the value of the MSS is within the 2 bit range (0-3). If the cookie meets both integrity controls, it is considered valid, and the connection can be accepted.

Table 2.1 – Parameters of the Linux implementation

Parameter	Description
K_1, K_2	Secret keys
IP_s, IP_d	Source and destination IP addresses
$Port_s$, $Port_d$	Source and destination ports
ISN_s, ISN_d	Source and destination initial sequence numbers
ACK	Acknowledgement number
SEQ	Sequence number
MSS	2 bit index of the client's Maximum Segment Size
Count	32 bit minute counter
Hash()	32 bit cryptographic hash

2.4 System Design

DDOS is one of the former and the most common threats that is growing in size and frequency in the networks. Accordingly, it is considered among the major attacks that exploit characteristics of the cloud environment [11], [18], [15]. The good capabilities of SDN, such as traffic examination based on software, centralized control and dynamic network reconfiguration may greatly improve the detection and mitigation of DDoS attacks in cloud computing environment. Additionally, the SDN data plane offers significant abilities, such as stateful memories that can keep track of network status, which can be useful in detecting network anomalies. In addition, it has powerful high throughput packet processing and analysis capabilities that can be used to defend against denial of service attacks. This capacity meets our goal of enabling detection and mitigation of overwhelming traffic attacks in the cloud system.

In this section, we introduce and present the system architecture of our proposed approach 'TCP SYN flooding defensive mechanism' which exploits the capabilities of SDN data plane using the network programming language P4 (Annex. A).

2.4.1 System Architecture

The objective of our proposed SYN flooding defensive system is to activate SDN data plane with clever and advanced functions to detect and mitigate SYN flood attacks in cloud environment. To achieve this aim, we extend the Data Plane (DP) with two modules: statistical attack detection module and mitigation and countermeasures module, as we can see in the overall architecture (Figure. 2.4). The proposed modules will be developed using python and P4 languages. The latter enables us to customize the process pipeline of incoming packets while creating specific parsers, flow tables, actions, and flow controls.

A. Statistical Attack Detection Module

The proposed detection mechanism activates the data plane to detect if cloud system is under flooding attacks at an early stage. The objective of deploying this module is the detection of overwhelming traffic and the prevention of an occurrence

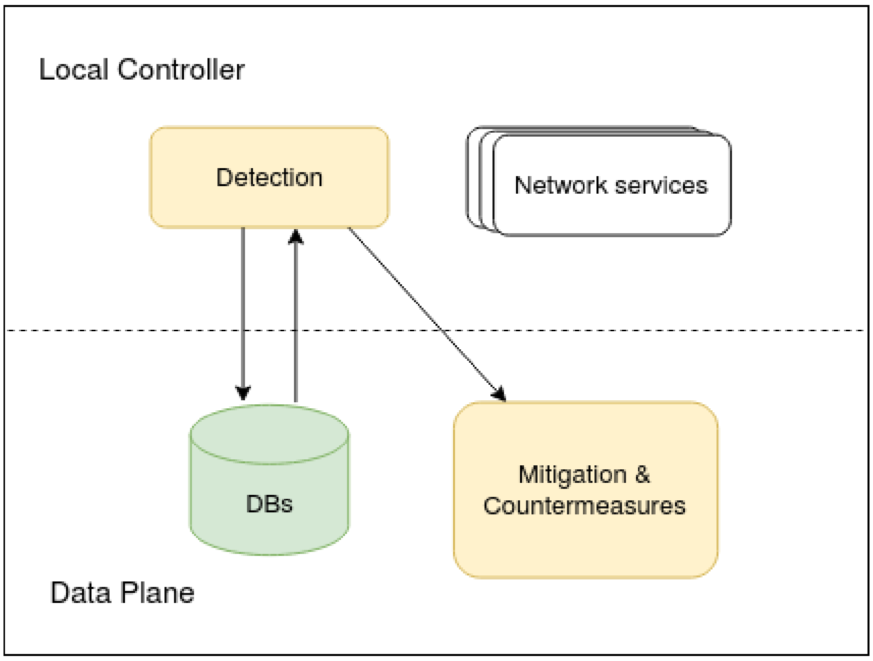


Figure 2.4 – Dynamic defense mechanism architecture

of flood attacks using the proposed mitigation application (Figure. 2.6). Therefore, the inspection and mitigation functions are only activated when flooding traffic is detected, rather than continually activating them. The detection mechanism uses the Cumulative SUM (CUSUM) (Subsection. 2.1) as a change point detection algorithm to detect the appearance of an unexpected change of the system traffic volume. It realizes this by modeling the normal behavior of the network traffic, based on the statistics collected in the data plane, and by reporting significant deviations from this behavioral model as anomalies, as described in Algorithm. 1 and shown in Figure. 2.5.

Statistical Data Collection

The SYN flooding attack is detected by examining the TCP SYN and FIN pairs' behavior. The SYN and FIN packets delimit the start (SYN) and end (FIN) of each TCP connection. That is mean, an appearance of a SYN packet results in the eventual return of a FIN packet in the normal condition. However, the RST packet violates the SYN–FIN pair, for any RST that is initiated to abort a TCP connection. According to the specification of TCP/IP protocol, in normal operation, a FIN (RST) is paired with a SYN at the end of data transmission [91].

In the normal condition, the difference between the number of SYNs and FINs (RSTs) is very small, as compared to the total number of TCP connection requests. Under a flooding attack, the difference between the collected number of SYNs and FINs (RSTs) will raise dramatically and remain significant throughout the entire flood period. Therefore, the appearance of a large difference between the number of SYNs and FINs (RSTs) for minutes or tens of seconds demonstrates a SYN flooding attack. During the same time period, the number of FINs (RSTs) remains largely unchanged. Thus, it remains interesting to focus on the SYNs collection which will be much larger than the FINs (RSTs) packets during the flooding attack period.

In the P4-enabled switch, there are options to maintain information across packets in real time, using stateful memories such us counters, meters and registers [4]. In our case, we use P4 counters to measure the number of incoming packets with SYN flag. We measure also FIN and RST packets to show the coherent synchronization between SYN and FIN (RST) packets. The maintained statistics activate the implemented detection and mitigation modules to realize the existence of flooding attacks and decide on adaptive countermeasures.

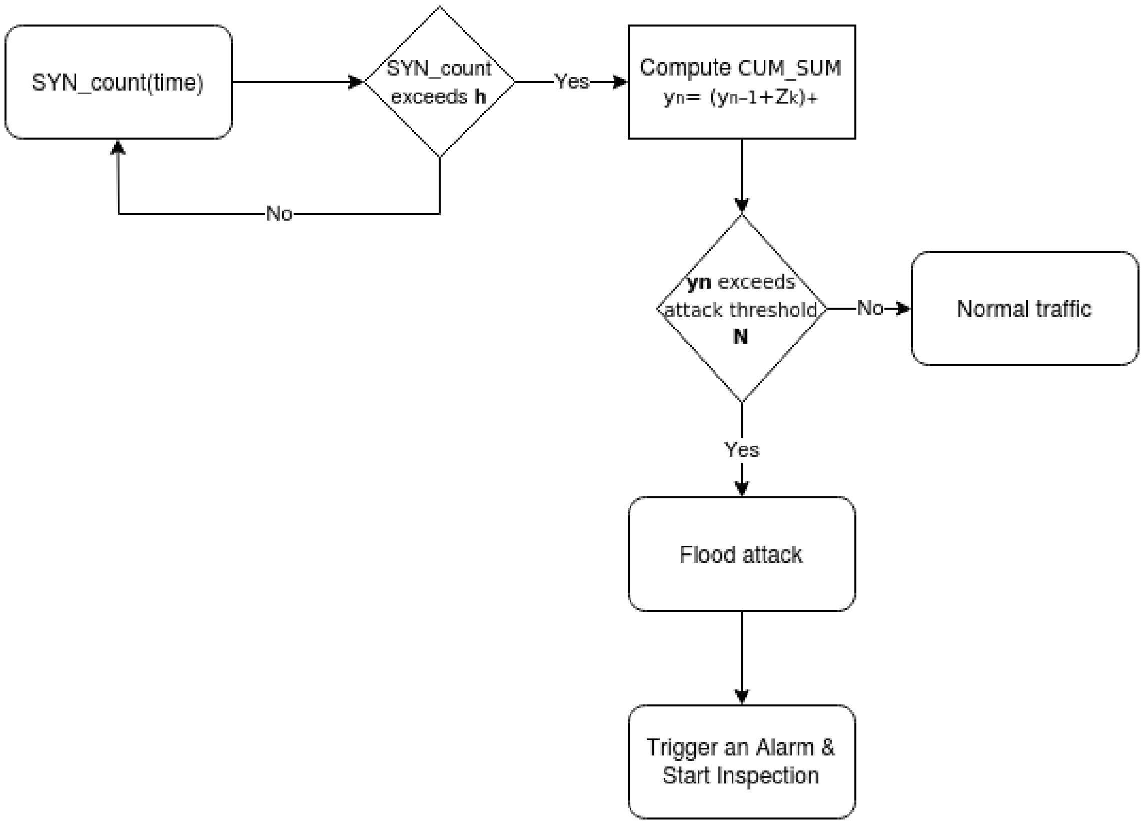


Figure 2.5 – Detection Mechanism Diagram

Detection mechanism

From the created P4 counters we get the number of SYN,FIN, and RST packets every time period (t0). Recent Internet traffic measurements have shown that most of TCP connections last 12–19 seconds [94], so we set (t0) to 20 seconds and the sampling time of FIN(RST) (t1) to 10 seconds. As mentioned above, we measure and recover FIN and RST packets to demonstrate the coherent synchronization between SYN and FIN (RST) packets. The detection mechanism use the SYN volume measurements to detect the flooding attacks, it is described in Algorithm.1.

B. Mitigation and countermeasure Module

When flooding traffic is detected the mitigation system starts its function (Figure. 2.6). It enables the DP to inspect the new incoming packets, which not exist in flow table, classify the benign from flooding packets, and activate the adaptive countermeasures, with a minimum switch-controller communication.

Algorithm 1: Detection algorithm

```
1  The detection system recovers the number of received SYN packets from P4
     counters every time period (t0), where t0=20 seconds. Every time period
     t0, the counter is put to rest to restart counting.
   input  : SYNcount, h, N
   output: Z, y
2  $SYN_{count} = \{0\}$;
3  $h = 10000$;
4  It retrieves the number of collected SYN packets, defined as $Z_n$, every time
     period (t0) and checks if $Z_n$ exceeds the abnormal traffic network
     threshold $h$ ($h$ defined as change point),
5  while $Z_n < h$ do
6      $Z \leftarrow SYN_{count}$;
7  Compute the Cumulative SUM $y_n$;
8  In this time, defined as starting time $k$, the detection system computes the
     Cumulative SUM $y_n$ by accumulating the $Z_k$ during a time interval, with
     the formula,
9  $y_n = (y_{n-1} + Z_k)^+$;
10 Definition of the flooding attack threshold $N$,
11 $N = 1$;
12 It compares the Cumulative SUM $y_n$ with the flooding threshold $N$,
13 if $y_n > N$ then
14     flooding attack;
15 else
16     Normal traffic;
```

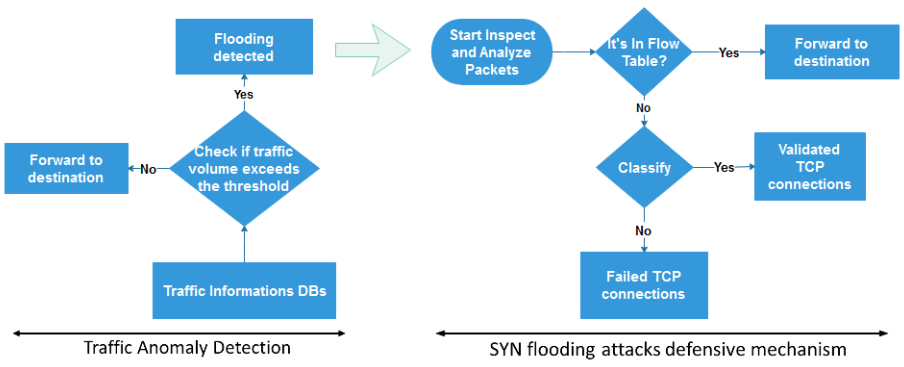

Flooding detected
Yes
Check if traffic volume exceeds the threshold
No
Forward to destination
Traffic Informations DBs
Start Inspect and Analyze Packets
It's In Flow Table?
Yes
Forward to destination
No
Classify
Yes
Validated TCP connections
No
Failed TCP connections
Traffic Anomaly Detection
SYN flooding attacks defensive mechanism

Figure 2.6 – Dynamic defense flowchart against SYN flooding attacks

In first time, DP checks if the packet exists in the flow table, if so, the packet will be immediately forwarded to the target server. Otherwise, the DP initiates a classification stage to classify the validated TCP sessions from failed ones (i.e., half-open SYN attacks or invalidated TCP sessions). We detail the classification stage in (Figure. 2.7). When the DP receives a TCP SYN/RST/FIN packet, it checks whether it is a SYN packet. If so, it increments the counter of the access table (i.e

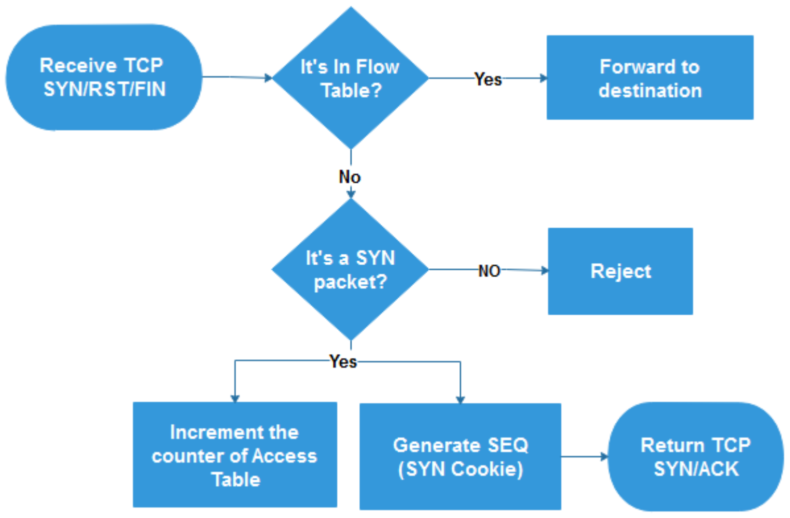


Figure 2.7 – Handling TCP SYN packets chart

contains information on all new TCP connection attempts) and generates a sequence number (i.e cookie) for this packet with hash functions (as shown in Subsection. 2.3.3) and sends back a SYN-ACK packet with a pre-generated cookie. If the packet is not a TCP SYN packet (i.e., TCP FIN or TCP RST), it is rejected. If the DP

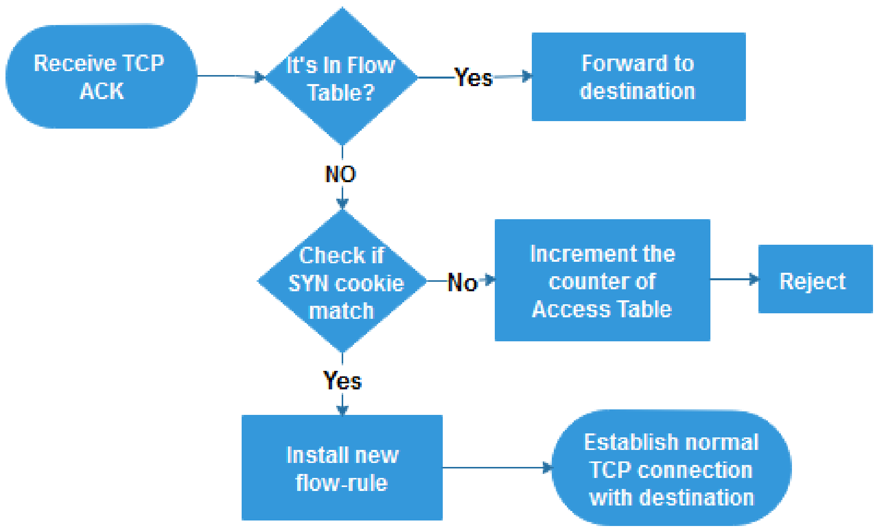

Receive TCP ACK
It's In Flow Table?
Yes
Forward to destination
NO
Check if SYN cookie match
No
Increment the counter of Access Table
Reject
Yes
Install new flow-rule
Establish normal TCP connection with destination

Figure 2.8 – Handling TCP ACK packets chart

receives an ACK packet, as shown in Figure. 2.8, it checks its TCP sequence number against the cookie value that was encoded in the SYN-ACK packet (as depicted in Subsection. 2.3.3). If the TCP connection is validated (i.e ACK packet contains the appropriate SYN cookie) the TCP handshake is established immediately between client and destination server. During accomplishment of the TCP handshake, the DP add the new rule in flow table to allow the future TCP requests arrived from this source.

2.5 Simulation

2.5.1 Environment

We use the Behavioral model-bmv2 framework which supports the P4 software switch, with v1model architecture. This reference implementation covers P4.16 specification version 1.2.0 [4], and it functions as the DP. The bmv2 framework can be used for two targets `simple_switch` (i.e., this target is a software switch, running on a general-purpose CPU, such as Intel/AMD/etc., that can execute `P4_-14` and `P4_16` programs) and `simple_switch_grpc`. The difference between both of the targets is that `simple_switch_grpc` can also accept TCP connections from a controller, where the format of control messages on this connection is defined by the P4 Runtime API specification. The P4 Runtime API is a control plane specification for controlling the DP elements of a device or program defined by a P4 program. P4c is used as a compiler; it compiles the P4 program (.p4) into the JSON file to be implemented on the P4 software switch, and defines the tables and actions, in Protobuf format(), existing in P4 program to be populated by the controller.

2.5.2 Implementation

Our focus here is the P4 software-based implementation, used to perform advanced security functions inside the switch, unlike the OpenFlow-enabled switch which requires hard and heavy changes to extend the data-plane. Our mechanism is based upon the Abstract forwarding model of P4 (Figure. 1.6) which consists of: **headers** describe the fields and their sizes of each header within a packet, **parser** defines the permitted header sequences within packets, **control flow** organizes the layout of match action tables within ingress and egress pipeline, and the packet flow through the pipeline, **match action tables** associate user-defined keys with actions, and **stateful memories** counters, meters, and registers are used to store information across packets (Annex. A).

A. Detection Application

We develop a detection application, in python language, to periodically retrieve statistics from connected switches and check if there is an overwhelming traffic attack that can overload the resources of network or cloud servers and possibly cause a SYN flood attack in cloud system.

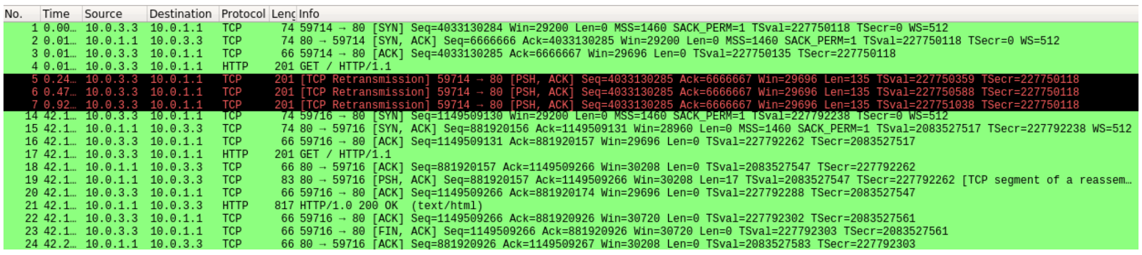
Our application creates a timer function *timer_function* which enables the local controller to send requests to the connected P4 switches and get the flow statistics (TCP SYN requests) every time period (20 seconds) using P4 Runtime interface. Statistics are retrieved from P4 counters which are already defined in our P4 program. P4 counters are programmed to count new incoming packets or TCP SYN requests which are useful information to detect or prevent the occurrence of SYN flood attack. At the same time as the count is recovered, the counter is put to rest to restart counting, and the detection application checks whether the maintained number of SYN requests exceeds the defined abnormal network traffic threshold h

10000 (500/second). IF the SYN_count exceeds the threshold, the system starts calculating the Cumulative SUM y_n during the time interval. Then the system checks whether y_n exceeds the flooding threshold N 1, if it is exceeded, the system triggers an alarm and automatically activates the mitigation application to start inspecting and classifying the new incoming packets, as shown in Figure. 2.5.

B. Inspection and Mitigation Application

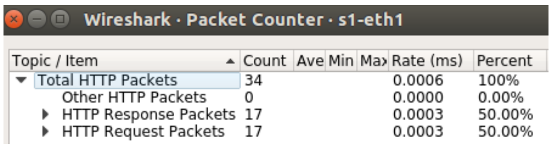
We develop mitigation application, in P4 network programming language, which describes how packets are analyzed and classified benign from flooding packets, and which defines the adaptive actions and countermeasures to perform. Below we describe the process of the mitigation application:

- We add some headers to define the required TCP fields and TCP options,
- we modify the parser to extract some specific fields/headers which look important to control the TCP three-way handshake.
- We create a counter to count the incoming packets with SYN flag. Optionally we create counters to count the incoming packets with FIN and RST flags to show the coherent synchronization between SYN and FIN (RST) packets.
- We add the **ipv4_lpm()** match action table to handle the incoming packet, it checks if the packet exists in the flow table, if so, then it forwards it to the target server using **forward_action()** action, if it does not exist, the action **send_to_cpu()** is taken to forward the SYN packet to the controller for adding the requested flow entry.
- When the detection application detects the occurrence of a SYN flood attack, it performs the following mitigation actions:
  - The first **return_SYN_ACK()** match action table allows the mitigation system to respond to the SYN packet by sending a `SYN_ACK` packet with a pre-generated sequence number (Subsection. 2.3.3),
  - The second **ACK_verify()** match action table enables the system to check and verify the validity of the sequence number existing in the received ACK packet.
- If the sequence number of the ACK packet is validated the action **send_to_cpu()** is performed to forward the validated ACK packet to the controller for adding the requested flow entry.
- The P4 Runtime controller learns from the received ACK packet to build the requested flow entry, and then write it on the DP for handling the incoming packets from the same flow.
- Optionally, the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client to enable him to re-establish the connection directly with the server.



```
No.  Time   Source    Destination Protocol Leng Info
   1 0.00…  10.0.3.3  10.0.1.1    TCP        74 59714 → 80 [SYN] Seq=4033130284 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=227750118 TSecr=0 WS=512
   2 0.01…  10.0.1.1  10.0.3.3    TCP        74 80 → 59714 [SYN, ACK] Seq=6666666 Ack=4033130285 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=227750118 TSecr=0 WS=512
   3 0.01…  10.0.3.3  10.0.1.1    TCP        66 59714 → 80 [ACK] Seq=4033130285 Ack=6666667 Win=29696 Len=0 TSval=227750135 TSecr=227750118
   4 0.01…  10.0.3.3  10.0.1.1    HTTP      201 GET / HTTP/1.1
   5 0.24…  10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Seq=4033130285 Ack=6666667 Win=29696 Len=135 TSval=227750359 TSecr=227750118
   6 0.47…  10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Seq=4033130285 Ack=6666667 Win=29696 Len=135 TSval=227750588 TSecr=227750118
   7 0.92…  10.0.3.3  10.0.1.1    TCP       201 [TCP Retransmission] 59714 → 80 [PSH, ACK] Seq=4033130285 Ack=6666667 Win=29696 Len=135 TSval=227751038 TSecr=227750118
  14 42.1…  10.0.3.3  10.0.1.1    TCP        74 59716 → 80 [SYN] Seq=1149509130 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=227792238 TSecr=0 WS=512
  15 42.1…  10.0.1.1  10.0.3.3    TCP        74 80 → 59716 [SYN, ACK] Seq=881920156 Ack=1149509131 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=2083527517 TSecr=227792238 WS=512
  16 42.1…  10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509131 Ack=881920157 Win=29696 Len=0 TSval=227792262 TSecr=2083527517
  17 42.1…  10.0.3.3  10.0.1.1    HTTP      201 GET / HTTP/1.1
  18 42.1…  10.0.1.1  10.0.3.3    TCP        66 80 → 59716 [ACK] Seq=881920157 Ack=1149509266 Win=30208 Len=0 TSval=2083527547 TSecr=227792262
  19 42.1…  10.0.1.1  10.0.3.3    TCP        83 80 → 59716 [PSH, ACK] Seq=881920157 Ack=1149509266 Win=30208 Len=17 TSval=2083527547 TSecr=227792262 [TCP segment of a reassem…
  20 42.1…  10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509266 Ack=881920174 Win=29696 Len=0 TSval=227792288 TSecr=2083527547
  21 42.1…  10.0.1.1  10.0.3.3    HTTP      817 HTTP/1.0 200 OK  (text/html)
  22 42.1…  10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [ACK] Seq=1149509266 Ack=881920926 Win=30720 Len=0 TSval=227792302 TSecr=2083527561
  23 42.1…  10.0.3.3  10.0.1.1    TCP        66 59716 → 80 [FIN, ACK] Seq=1149509266 Ack=881920926 Win=30720 Len=0 TSval=227792303 TSecr=2083527561
  24 42.2…  10.0.1.1  10.0.3.3    TCP        66 80 → 59716 [ACK] Seq=881920926 Ack=1149509267 Win=30208 Len=0 TSval=2083527583 TSecr=227792303
```

Figure 2.9 – Classification of successful TCP connection from flood attack: How the mitigation switch traits the legitimate packets



Topic / Item	Count	Ave	Min	Max	Rate (ms)	Percent
Total HTTP Packets	34				0.0006	100%
Other HTTP Packets	0				0.0000	0.00%
HTTP Response Packets	17				0.0003	50.00%
HTTP Request Packets	17				0.0003	50.00%

Figure 2.10 – Wireshark statistics: HTTP packet counter in the Web server

2.5.3 Use case

We use Mininet to emulate our test bed network architecture, as shown in Figure. ??. Mininet is implemented in conjunction with P4 to get Mininet-P4 environment; mininet topology that take in consideration P4 programs. Mininet generates the required topology while creating a default local controller which is responsible to populate switch flow tables, switches where each switch is running the P4 program, and hosts with IP configuration. We generate spoofed SYN flood attack using hping3, with various IP source, in switch S1. Normal traffic is generated using the open python program Scapy and it is generated in switch S2. The targeted HTTP server is installed in switch S3.

The Figure. 2.9 shows a new client, installed in S2, which sent an HTTP request targeted the web server (10.0.1.1). Since the client is new, the proper rule doesn't exist in flow-table, the mitigation switch will respond with a SYN-ACK packet and then check the validation of the received ACK packet (see the three first lines). Then, the validated ACK packet is transmitted as a packet in message to the controller to install the appropriate flow-entry. Then, the client (10.0.3.3) connects directly to the web server and get the requested data, as seen in the lines (14-24).

In this scenario, we measure the delivered packet rate of benign clients under network saturation attack.

The Figure. 2.12 indicates a spoofed SYN flood attack targeted the web server. In this case, the mitigation switch intercepts and responds to the received SYN requests, validate the successful TCP connections and ignore the failed one. Even though the network experiences several flood attacks, the web server is available because it only receives the authenticated TCP connections, and can process and respond to the HTTP requests from legitimate clients, as shown in Figure. 2.10.

To further show the impact of saturation attacks on normal traffic in detail, we vary the packet sending rate of the spoofed SYN flood attack from 0 to 1000 per second, and at the same time, we send the HTTP requests from 15 benign clients. The test results are shown in Figure. 2.11.

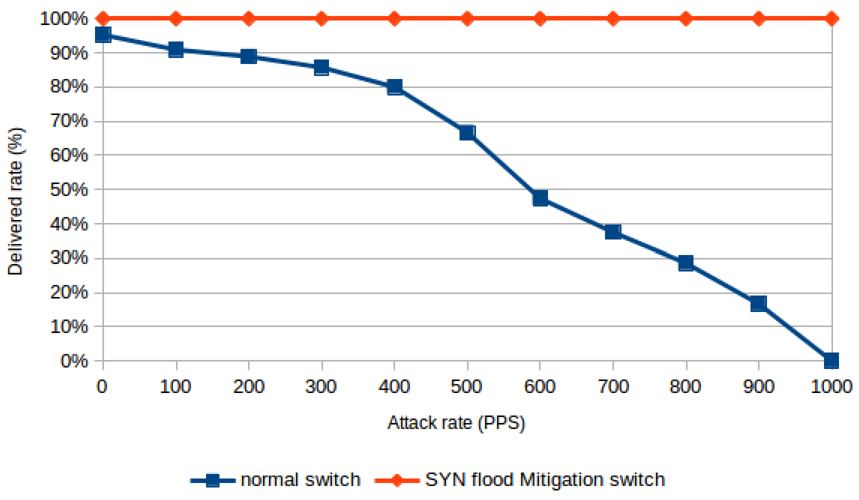
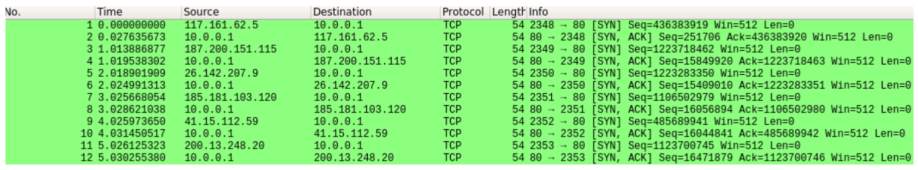


Figure 2.11 – Percentage of successfully delivered packets to the HTTP server from benign clients



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	117.161.62.5	10.0.0.1	TCP	54	2348 → 80 [SYN] Seq=436383919 Win=512 Len=0
2	0.027635673	10.0.0.1	117.161.62.5	TCP	54	80 → 2348 [SYN, ACK] Seq=251706 Ack=436383920 Win=512 Len=0
3	1.013886877	187.200.151.115	10.0.0.1	TCP	54	2349 → 80 [SYN] Seq=1223718462 Win=512 Len=0
4	1.019538302	10.0.0.1	187.200.151.115	TCP	54	80 → 2349 [SYN, ACK] Seq=15849920 Ack=1223718463 Win=512 Len=0
5	2.018901909	26.142.207.9	10.0.0.1	TCP	54	2350 → 80 [SYN] Seq=1223283350 Win=512 Len=0
6	2.024991313	10.0.0.1	26.142.207.9	TCP	54	80 → 2350 [SYN, ACK] Seq=15409010 Ack=1223283351 Win=512 Len=0
7	3.025668054	185.181.103.120	10.0.0.1	TCP	54	2351 → 80 [SYN] Seq=1106502979 Win=512 Len=0
8	3.028621038	10.0.0.1	185.181.103.120	TCP	54	80 → 2351 [SYN, ACK] Seq=16056894 Ack=1106502980 Win=512 Len=0
9	4.025973650	41.15.112.59	10.0.0.1	TCP	54	2352 → 80 [SYN] Seq=485689941 Win=512 Len=0
10	4.031450517	10.0.0.1	41.15.112.59	TCP	54	80 → 2352 [SYN, ACK] Seq=16044841 Ack=485689942 Win=512 Len=0
11	5.026125323	200.13.248.20	10.0.0.1	TCP	54	2353 → 80 [SYN] Seq=1123700745 Win=512 Len=0
12	5.030255380	10.0.0.1	200.13.248.20	TCP	54	80 → 2353 [SYN, ACK] Seq=16471879 Ack=1123700746 Win=512 Len=0

Figure 2.12 – How our mitigation system reacts to spoofed SYN flood attack

With SYN flood mitigation switch: The SYN flood packets are blocked by the mitigation switch and so, only the validated TCP connections are transmitted to the controller for writing the new flow-rules. In this case, web server will receive only the SYN requests from benign clients. As shown in Figure.2.11, the packet rate of benign clients are `100%` delivered to the web server, even while the network is under a severe network saturation attack.

With normal switch: All SYN requests, from the attacker, are transmitted to the controller as packet-in messages and thus, the controller is saturated and so unable to process the new packets from legitimate clients, which explains the packet rate from benign clients are nearly `0%` delivered.

2.5.4 Evaluation & Comparison

The main security features and strengths of our proposed framework are:

- `Useful DBs:`
  It exploits the stateful memories (i.e., stateful memories keeps track of the network state) capability of the SDN data plane using P4 language to realize customized traffic information DBs at the switch level. These DBs are useful for any kind of network security solutions to monitor and analyze network traffic and detect suspicious activities.

- `Early Detection:`
  It activates the detection of the overwhelming traffic attacks at the data plane

without any external and dedicated software application or hardware appliance. It realizes this while performing the Cumulative SUM(CUSUM) algorithm which enables the switch to detect the appearance of an unexpected change of the system traffic volume. In result, it prevents and reduces the risk of DDoS attacks in cloud servers.

- `Early Classification:`
  It rolls SYN cookie techniques closer to the switch to analyze and classify the legitimated TCP sessions from failed ones (i.e., half-open SYN attacks or invalidated TCP sessions) with a minimum switch-controller communication. In addition, it activates the SYN cookie technique only when the rate of TCP SYN requests reached a defined adaptive threshold, rather than enabling them constantly.

- `TCP session Management:`
  It stores information about all new TCP connection attempts including succeeded and failed connections which remain useful for detecting various attacks like network scanning attack.

- `Dynamic Network Reconfiguration:`
  It can activates the data plane with dynamic and reactive mitigation actions (i.e., block, reject, modify, redirect traffic, and add a new rule) against different security attacks.

- `Defense other protocols-based attacks:`
  It enhances the resilience against TCP SYN Flooding attacks and it may also be used to defend attacks based on HTTP, SMTP, and others protocols. This can be done while using specific SYN cookie methods, such as HTTP Redirect and TCP Safe Reset.

Based on our study, most of the existing DDoS mitigation mechanisms [18, 16, 19, 20] are designed at the high-level SDN application plane with the involvement of the SDN controller in each operation to detect and mitigate DDoS attacks. Therefore, the required communication channel between the control and data plane of SDN creates a potential bottleneck that impacts the network performance and restricts its scalability and reactivity. In comparison to these solutions, our proposed mechanism enables the data plane to detect the network saturation attack with a minimum communication between the SDN controller and data plane. In result, the switch may prevent and reduce the risk of DDoS attacks in downstream cloud servers at an early stage.

There are few works [69] [78] which perform SYN cookie technique at the SDN data plane to defeat TCP SYN flooding attacks. However, these approaches enable SYN cookie all the time and for each packet which limits the responsiveness of the system. Inspiring by the Linux kernel method, which automatically enables the SYN cookie only when the SYN queue is full, our framework activates the SYN cookie technique only when the rate of TCP SYN requests reached the defined adaptive threshold, rather than enabling them constantly.

In comparison to some existing solutions, our framework implements simple method functionalities over SDN switches rather than complex methods such as machine learning or deep learning which require a high memory and processing requirements.

2.6 Conclusion

Although the existing SDN-based security solutions offered significant benefits for a cloud environment, they can cause a network bottleneck that can be exploited by DDOS attacks. Our proposal addressed how to defend SYN Flooding attacks at the switch level using P4 programs, so the switch may perform an early prevention and mitigation of overwhelming traffic attacks in the downstream cloud servers. The proposed approach allowed the data plane to build databases of statistical information and perform traffic anomaly detection and SYN flood defense techniques. As a result, the switch could prevent traffic anomalies, mitigate SYN flooding attacks, and then decide on adaptive countermeasures.

Our solution attempts to defend cloud servers from SYN flood attacks using capabilities of SDN data plane, however SDN architecture itself suffers from such attacks. This challenge motivates us to design a security solution to first protect the centralized SDN controller and the SDN data plane from DDoS flood attacks, and to achieve an efficient networking system that can resist DDoS flooding attacks. In the next chapter, we will present the new proposed security solution 'DDoS flooding attack mitigation in Software Defined Networks' which aims to protect SDN components and the downstream servers.

Chapter 3

Secure and Resilient SDN with P4 Programmable Data Plane

3.1 Introduction

Software-Defined Networking (SDN) is an emerging network architecture that enables dynamic and efficient programmability, management, and provision of the networks. This helps managers control the entire network systematically and globally, regardless of the underlying network infrastructure. SDN capabilities, such as network-visibility, centralized control, network programmability, software-based traffic analysis enable IT administrators to implement easily in-network security functions, such as firewall, intrusion detection system (IDS), intrusion prevention system (IPS), etc. In this sense, many research works developed SDN-based frameworks for enhancing network-security in smart grids, IoT, cloud, etc [95, 96, 97]. Although SDN features help in managing and protecting large networking systems against various threats, SDN itself suffers from both the present attacks and new security issues. Due to the centralized controller and the network programmability of SDN, new security challenges emerged including malicious applications, distributed denial of service (DDoS) attacks, Fraudulent flow rules, flow-table overloading attacks, etc (see Section. 1.3).

DDOS is one of the former and the most common attacks which is increasing in size and frequency. In the past year, the cybersecurity vendor Akamai recorded hundreds of DDoS attacks per week. Recently, Kaspersky Labs observe an 84 percent rise in the number of DDoS attacks during the first three months of 2019 [98].

As presented in the literature, and as we discussed in Subsection. 1.4.3, DDoS attacks are considered among the major security threats that exploit SDN vulnerabilities. The SDN paradigm decouples the control plane from the data plane. The controller is the brain of the SDN architecture, responsible for implementing policies, retaining a global view of the network and providing a hardware abstraction layer to the control applications. Therefore, it can be considered a single point of failure if it is made unavailable by a DDoS attack. For example, an attacker can make the controller inaccessible by generating a series of new Transmission Control Protocol (TCP) SYN requests from distributed bots using spoofed Internet Protocol (IP) addresses, which involves the controller in a series of unnecessary processes. In this case, the controller will be saturated and thus unable to process legitimate requests. In addition, this type of attack also exhausts the bandwidth of the communication channel between the controller and the switches, as well as the flow tables of the switches with false flow rules [18], [8], [20].

This challenge motivated us to design and develop a lightweight and practical DDoS mitigation mechanism for protecting the SDN components and ensure a secure and efficient networking system that can resist DDoS flood attacks. The proposed mitigation system extends the Programmable Data Plane (PDP) with new modules including :

Classification and mitigation module allows to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures.

Control module activates the connection between the local controller to the switch(s), installs the classification and mitigation P4-programs on the switch(s), populates the defined flow-tables with flow rules, and describes how the controller interacts with the switch(s) and treats the incoming messages (packet-in).

Unlike the traditional fixed-function Openflow-enabled switch, P4-programmable switch is new concept to make the SDN network more flexible, dynamic, programmable, and scalable. In this research work, we choose to work with the P4-programmable switch to take full advantage of the SDN architecture and better secure their components.

The proposed mitigation mechanism will be developed in P4 programming language (Annex. A) and implemented using the behavioral model (bmv2) software switch [99] and Mininet emulator [100]. The simulation results indicate that the proposed defense system can efficiently tackle the DDoS flood attacks in the SDN architecture and also in the downstream servers.

Benefits of the proposed approach:

- Defend the data plane from saturation attacks using SYN cookie methods, as stateless technique, which doesn't require storing the state of TCP connections.

- Discharged the communication path between control and data planes by performing classification and mitigation functions at the switch level.

- Protect the centralized controller from the flooding attacks which affect their availability and scalability.

- Enhance the resistance of SDN architecture against DDoS flooding attacks.

The present chapter is organized as follows: In Section 3.2, we evaluate some of the existing research works which propose security solutions to detect and mitigate DDoS threats in SDN. We present the design and architecture of our approach "Secure and Resilient SDN with P4Programmable Data Plane" in Section 4.3. The simulation implementation and results are presented and evaluated in Section 3.5. Finally, we give our conclusion and perspectives in Section 4.4.

3.2 Related work

A large number of research projects proposed defense approaches against DDoS attacks using SDN [101, 102, 103, 96, 104] whereas, SDN architecture itself suffers from different kind of DDoS attacks. A few works address the potential challenges to mitigate DDoS attacks in SDN [105, 106, 107, 108, 82, 83]. Among which we present and analyze the following research works:

Mousavi et al. [105] exploited the SDN controller's broad network feature to protect SDN architecture from DDoS attacks. They proposed an entropy-based solution, that works within the controller, to detect DDoS attacks against SDN controllers. The proposed algorithm adds two functions to the controller; One is collecting the new incoming packets to the destination IP addresses into window size 50 and the other computes the entropy of each window and compares it to an experimental threshold. In [106] Tran et al. proposed the ODL-ANTIFLOOD solution which detects and mitigates the Controller-aimed DDoS flooding attacks. The proposed detection technique is developed based on the combination of entropy and packet in rate methods. The mitigation technique includes three steps: identify attack sources, mitigating the impact of attacks, and recovering SDN system after the attacks. Tank et al. [107] suggested an SDN-based Network Intrusion Detection System architecture . The proposed NIDS takes advantage of software-based traffic analysis and logical centralized control of SDN for detecting intrusion. They constructed a simple Deep Neural Network (DNN) for the intrusion detection system using the NSL-KDD Dataset. In [108] Li et al. applied the deep learning model to detect DDoS attacks in Software Defined Networking (SDN) environment using the ISCX data set. After the collection and analysis of network traffic feature information, the deep learning model is used for feature reduction and DDoS attack detection.

Their suggested solutions involve the controller in every operation to detect and mitigate DDoS attack, which may overload the control and data planes path and creates a potential bottleneck that impacts the network performance and restricts the scalability and reactivity of SDN controller. Moreover, the [107, 108] use complex method (i.e., deep learning) which requires a high memory and processing requirements.

Shin et al. [82] presented Avant-guard as a detection and prevention solution against the TCP SYN flooding attack. Avant-guard extends the data plane with the connection migration module which proxying the incoming TCP SYN packets and prevent the control plane from saturation attack. LineSwitch [83] is an improved Avant-guard solution. LineSwitch addressed the vulnerabilities and limitations of Avant-guard by proxying a minimum number of TCP SYN requests. The implementation of [82, 83] solutions requires complex design and several lines to add and modify to extend and customize the data-plane.

Recently, Afek et al. [7] have implemented different SYN cookie methods, as anti-spoofing mechanisms, in OpenFlow 1.5 using Open vSwitch (OVS) and P4 to protect downstream server. Their programming experience indicates that the implementation of novel applications which operate with new header fields in OpenFlow environment required complex design and several lines to add and modify compared to the programming in P4 switch. That experience confirm and validate for us the use of P4 programming language [4] and bmv2 software switch to implement our proposed SYN flood mitigation mechanism.

3.3 Selected Methods

In this contribution we attempt to protect SDN from spoofed DDoS attacks including SYN and UDP flooding threats. Against a spoofed SYN flood request we use the same methods as the first contribution (see Subsection. 2.3.2). Following [109], the same method (SYN flooding defense) of the first contribution can be applied to mitigate a spoofed DNS request in SDN.

UDP flooding defense

DNS spoofed attack is a type of DDoS attack, which attempts to exhaust the targeted server resources by sending a large amount of simultaneous of DNS requests. The majority of DNS traffic on the Internet goes through User Datagram Protocol (UDP), which is an unreliable and connection-less protocol. Thus, it is easy for hackers to spoof DNS/UDP messages and thus to overload the DNS server.

Although the majority of DNS traffic is routed over UDP, DNS can also be routed over Transport Control Protocol (TCP) connections. Therefore, when the mitigating switch receives a DNS UDP request from a client it has not authenticated before, it can force the client to repeat its request over TCP. The client establishes a TCP connection with the mitigating switch, using the conventional three-way handshake mandated by the TCP protocol. The mitigating system uses the handshake to verify the authenticity of the client using a cookie as described in Section. 2.3. To force the client to repeat its first UDP request in TCP, the TC bit (Truncated - Indicates that the UDP response is too long to be carried by UDP) in the DNS header should be set in the response.

This method is similar to the TCP-Reset and HTTP-Redirect methods, as described in Subsection. 2.3.2. As illustrated in Figure. 3.1, the switch responds to the original UDP request by setting the TC bit. Then after the TCP handshake is completed, it forwards the TCP DNS request to the controller that translates the TCP to UDP both for the request and response from the DNS server. Finally, the controller installs the adaptive flow rules to allow future DNS UDP requests from this authenticated client.

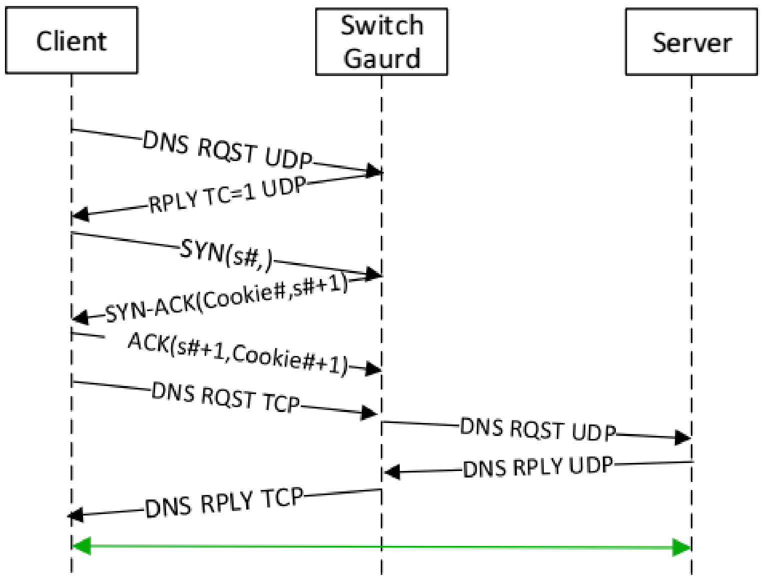


Figure 3.1 – DNS spoofed attack mitigation [7]

3.4 Proposed secure SDN architecture

3.4.1 Tools of the proposed SDN architecture

The proposed SDN architecture comprises Open Network Operating System (ONOS) controller [110] and P4 programmable data plane communicating through P4 Runtime API [111].

A. ONOS controller

The core concept of Software Defined Networking is separating the intelligence and control from forwarding components and concentrating the control of the network management and operation in a logically centralised SDN controller. Many SDN controllers exist, among which we list the most popular open source SDN controllers in industry and academia including the ONOS, OpenKilda, Ryu, OpenDayLight (ODL), Floodlight, NOX, POX, etc.

To build our experimental environment, we have selected ONOS controller among other SDN platforms for the following reasons:

- ONOS has built-in mechanisms for connecting/disconnecting elements while the controller is running. This allows a very flexible approach to adding functionality to the controller.
- It supports a full list of southbound interfaces including OpenFlow, P4 Runtime, NETCONF, RESTCONF, etc. In addition, ONOS offers the largest set of northbound interfaces with gRPC and RESTful APIs.
- The Open Network Operating System offers the flexibility to develop and implement new dynamic network functions with simplified programming interfaces such as P4Runtime API.

- It allows administrators to dynamically reconfigure the network and control network elements in real time.
- The scalability of ONOS make it highly available and resilient against failure which increases the customer user experience.

B. P4-Programmable data plane

Although the benefits of SDN architecture to improve the network management, the fixed-function of the standard Openflow-enabled switch offers a complex design to enable variations in data plane pipelines. Unlike the traditional fixed-function switch, P4-programmable switch is new concept to make the network more flexible, dynamic, programmable, and scalable. P4-programmable switch is based on programmable packet processing pipeline (see Subsection. 1.2.3). The data plane functionality is not fixed in advance but is defined by P4 programs. Programmers are able to design P4 programs for describing how packets are processed by the data plane and defining only the functionalities that are needed.

P4-programmable switch allows programmers to include only the necessary protocols in the code, which leads to more efficient systems than general purpose appliances. Moreover, it enables network equipment designers and end users to create and implement new protocols and features without depending on the providers of the specialized packet processing ASICs to implement custom algorithms.

In this research work we have used the programming protocol independent packet processors - P4 and P4Runtime API to run a programmable data plane (see Subsection. 1.2.3).

P4 is a programming language for describing how packets are processed by the data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router, or network appliance. P4 program is based upon an abstract forwarding model consisting of a parser, deparser, and a set of match-action table resources, divided between ingress and egress.

P4Runtime API has been designed and standardized by Google and Barefoot Networks. P4Runtime targets a remote controller and is defined using Protobuf (i.e.,serialization protocol) and gRPC (i.e.,RPC protocol). The main goal of a P4Runtime is to manage P4 objects at runtime and to dynamically provision switches with the appropriate P4 program. P4Runtime is a protocol-independent API, which allows vendors to adopt it without any complexity. It can be used on the existing fixed-function switches, enabling their forwarding behavior to be programmable (i.e., coding in P4 language).

3.4.2 Proposed mitigation approach

The goal of our proposed defensive mechanism is to activate the programmable data plane (PDP) with smart and advanced functions to mitigate DDoS flood attacks in SDN architecture. To reach this aim, we extend the PDP with a classification and mitigation module, developed in P4 language, to analyze and classify the new incoming packets and perform the adaptive countermeasures, as depicted in Figure. 3.2. In addition, we create a control module, designed in python language, which

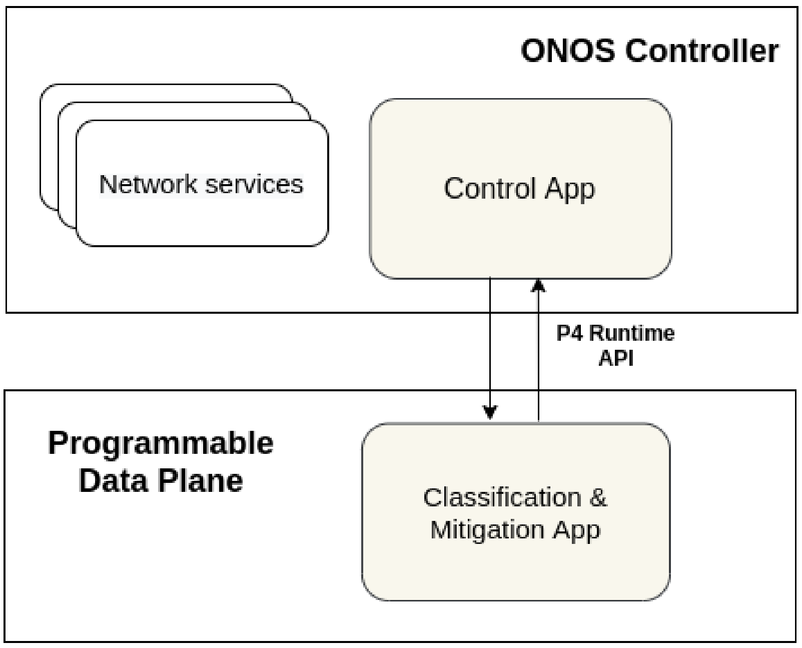


Figure 3.2 – Overall Architecture

will be implemented in the ONOS controller and interacts with P4 switch messages (i.e. packet_in). The controller is responsible for controlling the P4 switches and populating the flow tables through the P4 Runtime API in real-time. The proposed mitigation approach make the SDN architecture more resistant against DDoS attacks and ensure a secure and efficient networking system.

3.4.3 Classification and mitigation module

The classification and mitigation module allows the PDP to analyze and classify the new incoming packets. PDP will be able to classify the benign packets from the SYN flood attacks and to execute the adaptive countermeasures, as shown in Figure. 3.3.

In first, PDP checks if the incoming packet exists in the flow table if so, the packet will be immediately forwarded to the destination server. Otherwise, the PDP checks whether it is a TCP SYN or an ACK packet. If it is a SYN packet, the PDP returns a SYN-ACK packet with a pre-generated initial sequence number (ISN). Otherwise, if the packet is not a TCP SYN packet or an ACK packet, it is rejected.

Pre-generated sequence number or pre-generated cookie should contain some details of the initial SYN packet and its TCP options and it must also be unpredictable by attackers. Therefore, it is recommended to use cryptographic hashing functions in order to make the decoding of ISN more complicated. For this end, we select the Cyclic Redundancy Check 16 (CRC16) hash function to generate the hash values. Additionally, we apply the recommended Linux SYN cookie method to generate and validate the cookies [93].

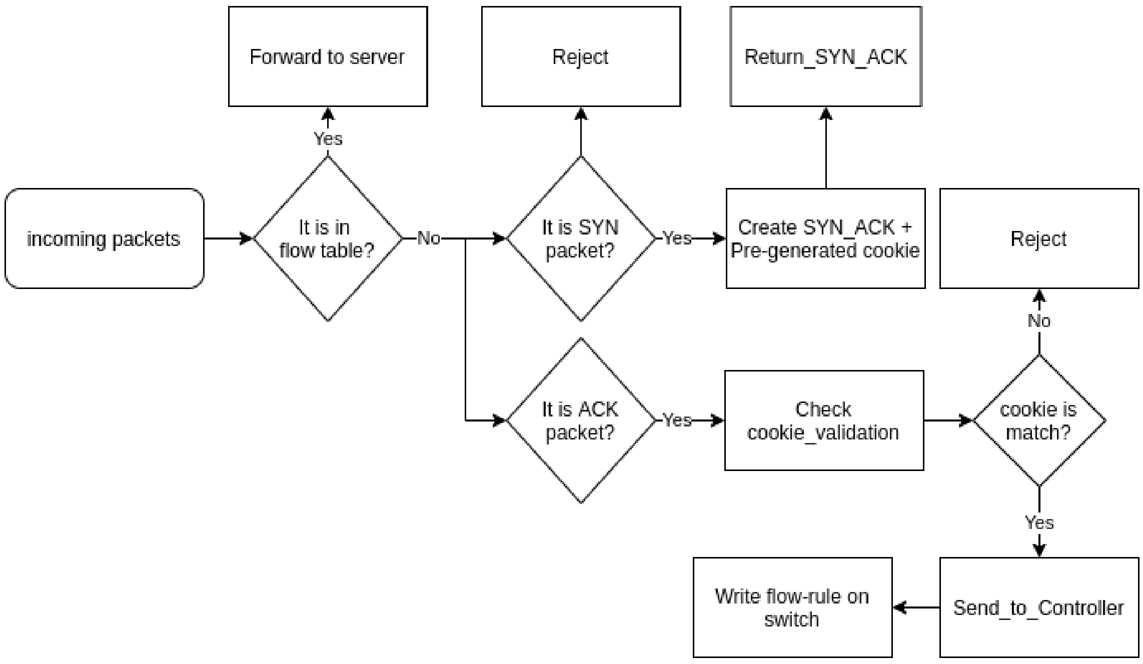


Figure 3.3 – TCP SYN flood defensive flowchart

Generation of cookies (ISN):

$$H_1 = hash(K_1, IP_s, IP_d, Port_s, Port_d) \tag{3.1}$$

$$H_2 = hash(K_2, count, IP_s, IP_d, Port_s, Port_d) \tag{3.2}$$

$$ISN_d(cookie) = H_1 + ISN_s + (count \times 2^{24}) + (H_2 + MSS) \mod 2^{24} \tag{3.3}$$

We use the CRC16 hash function to generate the hash values H_1 and H_2. The CRC16 hash function takes as inputs the IP addresses, ports, secret keys k_1 and k_2 and timestamp(count). Then, the destination sequence number (ISNd) is generated while computing the hash values, ISN of the source and max segment size (MSS). One the cookie is generated, we passe to create the SYN_ACK packet. This latter is generated simply by modifying the initial SYN packet to become a SYN-ACK packet (see Figure. 3.4).

To realize this, the following primitive steps are performed on the SYN packets that don't match any flow entry.

1. Swap the source and destination IP addresses.
2. Exchange the source and destination Ethernet addresses.
3. Swap the source and destination TCP ports.
4. Set the ACK bit in the TCP flags.
5. Increment the client-Seq number field by one.
6. Copy the incremented Seq number field to the ACK number field.

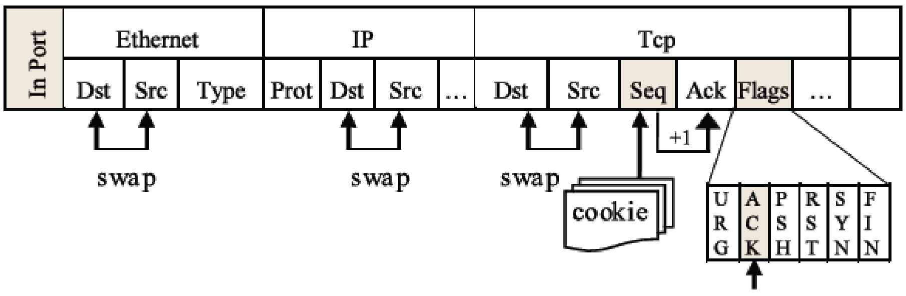


Figure 3.4 – Modifying SYN packet to become a SYN-ACK packet [7]

7. Write a pre-generated random cookie to the Seq number field.
8. Recalculate the IP/TCP checksum values (in P4).
9. Send back the SYN-ACK packet on the incoming port (on which the SYN-packet was received).

If the PDP receives an ACK packet, as shown in Figure. 3.3, it checks the validation of the TCP sequence number (i.e., is the ACK-1). For the validation of the cookie, we control two integrities: count(cookie) for checking the age of the cookie (must be lower to 2min), and MSS(cookie) for evaluating whether the value of the MSS is within the 2-bit range (0-3). If the cookie meets both integrity controls, it is considered valid, and the connection can be accepted.

Validation of cookies:

$$ISN_d = ACK - 1 \tag{3.4}$$

$$ISN_s = SEQ - 1 \tag{3.5}$$

$$count(cookie) = (ISN_d - H_1 - ISN_s)/2^{24} \tag{3.6}$$

$$MSS(cookie) = (ISN_d - H_1 - ISN_s) \mod 2^{24} - H2 \mod 2^{24} \tag{3.7}$$

If the TCP connection is validated the switch transmits the ACK packet as packet in message to the controller to write the required flow-entry. Optionally, the switch sends back a TCP-reset packet, with source IP of the original server, to the client in order to enable him to re-establish the connection directly with the server. In our system, we use a local controller that is responsible for controlling the P4 switches via the P4 Runtime API in real time.

3.4.4 Control module

We create a control module, in python language, to control the P4 data plane and populates the flow tables at runtime. The module connects the ONOS controller to the P4 switches, installs the classification and mitigation P4-programs on the switches, populates the defined flow-tables with flow rules at runtime, and describes how the ONOS controller treats the incoming messages (packet-in).

Once the controller receives a packet-in message, it learns details of the validated ACK packet to create the requested flow rule. It extracts the IP source, IP destination, MAC source and MAC destination addresses, and the source and destination ports. It adds the destination IP address and the corresponding MAC address to its memory, if it does not exist. It also checks for the existence of the source IP address and its corresponding Mac address, and adds them if they do not exist. Then the controller is able to write automatically the requested flow rule to the switch. Each flow rule is installed with a timeout value, both hard and idle timeouts, to avert the growth of rules in the switch flow-tables.

3.5 Simulation

3.5.1 Environment

We use the Behavioral model framework (bmv2) which supports the `P4` software switch, with `v1model` architecture. This reference implementation covers $P4_{16}$ specification version 1.2.0 [4], and it functions as the PDP. The bmv2 framework can be used for two targets `simple_switch` and `simple_switch_grpc`. The simple_-switch target is a software switch, running on a general-purpose CPU, such as Intel/AMD/etc., that can execute $P4_{14}$ and $P4_{16}$ programs. The simple_switch_grpc target accept also TCP connections from a controller, where the format of control messages on this connection is defined by the P4 Runtime API specification. The P4 Runtime API is a control plane specification for controlling the PDP elements of a device or program defined by a P4 program. P4c is a reference compiler for the P4 programming language, it compiles the P4 program (.p4) into the JSON file to be implemented on the P4 software switch, and defines the tables and actions, in Protobuf format(), existing in P4 program to be populated by the controller.

3.5.2 Implementation

Our focus here is the software-based implementation, used to direct the packet handling process inside the switch. Our implementation is based upon the P4 Abstract forwarding model (see Figure. 1.6) which consists of: **headers** describe the fields and their sizes, **parsers** define how headers are organized together and how to distinguish between them, **match action tables** associate user-defined keys with actions, **control flow** organizes the layout of match action tables within ingress and egress pipeline, and the packet flow through the pipeline, and **stateful memories** including counters, meters, and registers are used to store information across packets (see Subsection. 1.2.3).

The proposed classification and mitigation mechanism will be coded in P4 language to obtain P4 program that will be compiled, using the P4c compiler, and installed in the P4 switches. We have used v1model $P4_{16}$ architecture which already defines Ethernet and IP headers.

A. Classification and mitigation application:

Classification and mitigation application includes a set of P4 programs describing headers, parser, deparser, checksum verification and computation, ingress processing including match-action tables, and egress processing which describes how packet-in messages will be forwarded to the controller (CPU_port). We present below the designed components:

- We add other headers to define TCP fields and TCP options, controller port (CPU_port), and packet-in and packet-out.
- The parser is changed to extract TCP fields and options.
- The following ingress packet processing pipeline includes three match-action tables.

```
apply {
 ipv4_lpm.apply();

 if (hdr.tcp.flags == 02) {
    return_synack();
 }
 else if (hdr.tcp.flags == 10) {
    ACK_verify();
    if ((meta.meta.count_cookie <= 2) &&
    (0 <= meta.meta.mss_cookie) &&
    (meta.meta.mss_cookie <= 3)){
send_to_cpu();}
else{
drop(); }
}}
```

- The first one, named **ipv4_lpm()** which handles the incoming packet, checks the existing of the corresponding flow entry, and then forwards it to the target server. If the received packet doesn't match any flow entry, the classification stage is performed to classify the validated TCP connections from flooding packets.

```
action ipv4_forward(macAddr_t dstMac, egressSpec_t port) {
standard_metadata.egress_spec = port;
hdr.ethernet.srcMac = hdr.ethernet.dstMac;
hdr.ethernet.dstMac = dstMac;
hdr.ipv4.ttl = hdr.ipv4.ttl - 1;
}
```

- The second one, named **return_SYN_ACK()** which responds to the SYN packet by sending a `SYN_ACK` packet with a pre-generated sequence number (see Subsection. 2.3.3).

```
action return_syn_ack()
    {
//Timestamp
meta.meta.hash_count =
(bit<32>)standard_metadata.ingress_global_timestamp;
//MSS - Max Segment Size extracted from SYN packet
meta.meta.mss = (bit<32>)hdr.Tcp_option_ss.maxSegmentSize;

//H1 = hash(IP_s, IP_d, Port_s, Port_d, K1)
hash(meta.meta.hash_1, HashAlgorithm.crc16, (bit<16>)0,
{hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, hdr.tcp.srcPort,
hdr.tcp.dstPort, key_1 }, (bit<32>)65536);

//H2 = hash(IP_s, IP_d, Port_s, Port_d, K2, timestamp)
hash(meta.meta.hash_2, HashAlgorithm.crc16, (bit<16>)0,
{hdr.ipv4.srcAddr, hdr.ipv4.dstAddr, hdr.tcp.srcPort,
hdr.tcp.dstPort, key_2, meta.meta.hash_count }, (bit<32>)65536);

//ISN_d(cookie) = H1 + ISN_s +
(timestamp × 2^24)+(H2 + MSS) mod 2^24

meta.meta.cookie = meta.meta.hash_1+hdr.tcp.seqNo+
(meta.meta.hash_count*0x01000000)+
(meta.meta.hash_2+meta.meta.mss) & 0xffffff;
//SWAP fields
swap_add_mac(hdr.ethernet.srcMac, hdr.ethernet.dstMac);
swap_add_ip(hdr.ipv4.srcAddr, hdr.ipv4.dstAddr);
swap_port_nb(hdr.tcp.srcPort, hdr.tcp.dstPort);
        hdr.tcp.flags = 8w0x12;
hdr.tcp.ackNo =(bit<32>)(hdr.tcp.seqNo + 32w0x00000001);

//afect the hash_cookie to SeqNo
        hdr.tcp.seqNo = (bit<32>)meta.meta.cookie;
standard_metadata.egress_spec = standard_metadata.ingress_port;
    }
```

- The third one, named **ACK_verify()** which checks and verify the validity of the sequence number existing in the received ACK packet, as described in Subsection. 2.3.3.

```
action cookie_verify()
    {
//SWAP
meta.meta.ISN_d = hdr.tcp.ackNo-1;
meta.meta.ISN_s = hdr.tcp.seqNo-1;
//if count_cookie <=2 minutes and mss_cookie
is within the 2 bit range (0,1,2,3) the ACK is valid

//count(cookie) = (ISN_d - H1 - ISN_s)/2^24
meta.meta.count_cookie = (meta.meta.ISN_d - meta.meta.hash_1 - meta.meta.ISN_s)
/ 0x01000000;
//MSS(cookie) = (ISN_d - H1 - ISN_s) mod 2^24  H2 mod 2^24
meta.meta.mss_cookie = (meta.meta.ISN_d - meta.meta.hash_1 - meta.meta.ISN_s)
& 0xffffff - meta.meta.hash_2 & 0xffffff;
    }
```

- If the sequence number of the ACK packet is validated the action **send_-to_cpu()** is performed to forward the validated ACK packet to the controller for adding the requested flow entry. This is applied in the egress processing pipeline as we can see below:

```
apply {
    if (standard_metadata.egress_port == CPU_PORT) {
        hdr.packet_in.setValid();
        hdr.packet_in.ingress_port =
        (bit<16>)standard_metadata.ingress_port;
    } }
```

- The controller learns from the received ACK packet to build the requested flow entry, and then write it on the PDP for handling the incoming packets from the same flow.
- Optionally, the switch sends back a TCP-reset packet (i.e., with source IP of the original server) to the client to enable him to re-establish the connection directly with the server.

B. Control application:

The control application comprises a set of functions: the *write_ ACK_ reply* and the *main* functions. The function *write_ ACK_ reply* allows to install forwarding ip_v4 flow rules in P4 switch.

The main function *main* allows to:

- Create a switch connection object for P4 switches; this is backed by a P4Runtime gRPC connection. Also, dump all P4Runtime messages sent to switch to the given text files,
- Send master arbitration update message to establish the present controller as master, it is required by P4Runtime before performing any other write operation,
- Install the P4 program on the switches using *SetForwardingPipelineConfig.*

The proposed program allows controller to write forwarding rules manually and automatically. It writes automatically a flow entry while learning details of received packet-in messages. Classification and mitigaton application enables the data plane to forward only validated TCP connections to the controller which offloads the resources of control-data communication channel, controller and switches.

The command line below describes how to create manually a flow entry for packets coming from 10.0.1.1 and forwarding to 10.0.2.2 :

```
writeACKReply(p4info_helper, sw=s1, in_port=1,
src_ip_addr="10.0.1.1", dst_ip_addr="10.0.2.2",
dst_eth_addr="08:00:00:00:02:22", port=2)
```

To create automatically the requested flow rule, the controller learns details of the received and validated ACK packet. It extracts the IP source, IP destination, MAC source and MAC destination addresses, and the source and destination ports. It adds the destination IP address and the corresponding MAC address to its memory, if it does not exist. It also checks for the existence of the source IP address and its corresponding Mac address, and adds them if they do not exist. Then the controller is able to write automatically the requested flow rule to the switch.

The command line below describes how to add the Mac address source or destination in the memory if not exist, and create automatically the flow entry based on the extracted packet information and write it on the switch.

```
if pkt_eth_dst not in memo_rules[value]:
    writeACKReply(p4info_helper, sw=s1, in_port=value,
    dst_eth_addr=pkt_eth_dst, port=port_map[pkt_eth_dst])
    memo_rules[value].append(pkt_eth_dst)
```

3.5.3 Use case

The test environment of this experiment includes the ONOS controller, p4-enabled switch in which our mitigation program is implemented, P4 Runtime API used to control P4 objects and populate the existing match tables, machines for generating normal traffic. We simulate a real DDoS attack environment using the Hyenae tool. The latter can customize the generation of the packet traffic, in order to simulate different types of DDoS attack packets in the SDN network, such as SynFlood, UDPFlood, ARPFlood, PingofDeath, Smurf, etc. In our case, we use Hyenae to generate SynFlood and UDPFlood. We use mininet and P4 to create a software-based SDN network environment. Normal traffic is generated using the open python program Scapy.

Controller-aimed DDoS attack type affects not only the controller resources, but also the switch-controller communication channel and the flow table switches. To anticipate such attack, it is interesting to follow the linear growth of the packet in message rate as well as the performance of the controller under DDoS attack. Controller performance relates to its compute resources, including CPU utilization or processing power, memory, storage, time, bandwidth, etc. Here we focus on measuring controller CPU utilization and switch-controller bandwidth consumption in normal traffic and under DDoS attack.

A. CPU Utilization

Figures (3.5, 3.6) show the CPU utilization of controller in different states. In normal traffic, the CPU resource usage remains stable (approximately 20%) as seen at the beginning of the graph (Figure. 3.5). At 42 min (420 seconds), a flood attack was launched, which explains the sudden increase in CPU usage, as shown in Figure. 3.5, from 20% to almost 80% and continue to grow with attacks. Under DDoS attacks, without the protection system, the controller receives a large number of packet in messages and automatically the CPU usage increased. This level of consumption can cause a lot of trouble for the controller, because the remaining resource is too low to be used in the computation, even the controller can be shut down. With the mitigation system, in Figure. 3.6, during attacks, the controller only receives the validated packet in messages, so the CPU utilization remains stable as in normal traffic, approximately 20%. The result shows that the proposed mitigation application is a lightweight solution and can effectively save the controller computing resources.

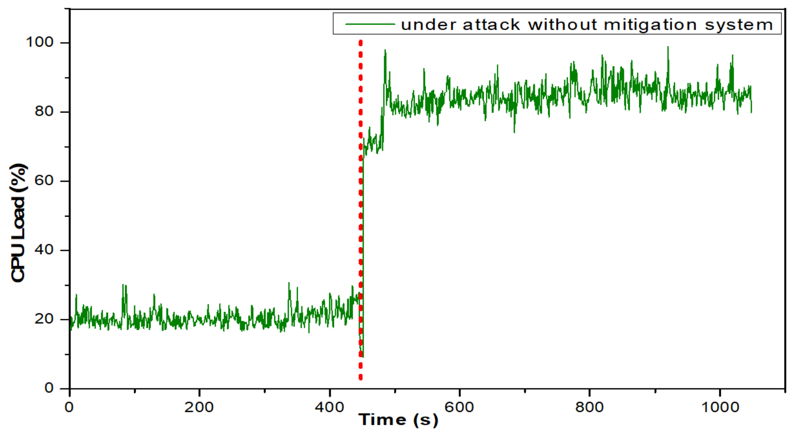


Figure 3.5 – CPU Utilization of Controller without mitigation system

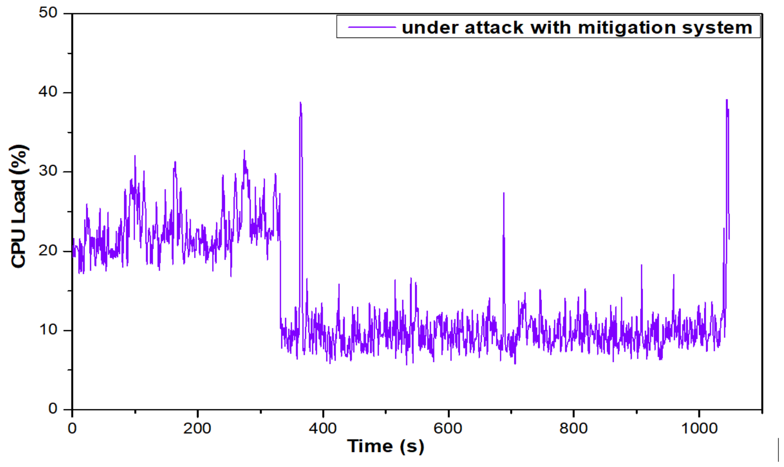


Figure 3.6 – CPU Utilization of Controller with mitigation system

B. Switch-to-Controller Bandwidth Consumption

Figures (3.7, 3.8) show the bandwidth usage between the switch and the controller in various states. In normal traffic, as illustrated at the beginning of the graph (Figure. 3.7), the number of packet in messages arriving to the controller is not large, due to the low intensity of table-miss packets in stable network environment. However, when the network is under attack, switch-controller channel bandwidth is nearly taken by nine thousands of malicious packet in messages sent from attackers, as seen in Figure. 3.7. As results, bandwidth saturation prevents legitimate traffic from reaching the controller. With the mitigation system, the attack traffic are classified and blocked at the switch level without any involvement of controller, therefore the switch-controller channel bandwidth usage remains stable and so available to respond to benign users, as shown in Figure.3.8.

The result shows that the proposed mitigation application is a lightweight solution as it is implemented and executed at the switch level and therefore does not consume controller resources. In addition, it can effectively save the computer resources of the controller, as well as the switching memory, the capacity of the

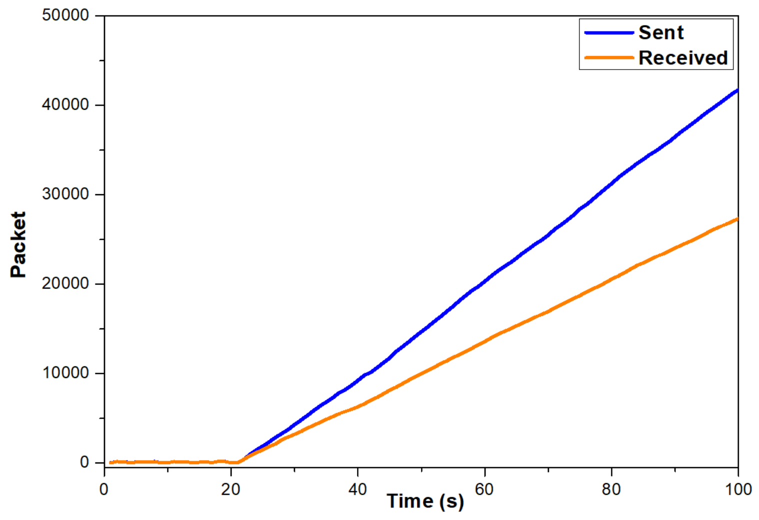


Figure 3.7 – Switch-to-Controller Bandwidth Consumption without mitigation system

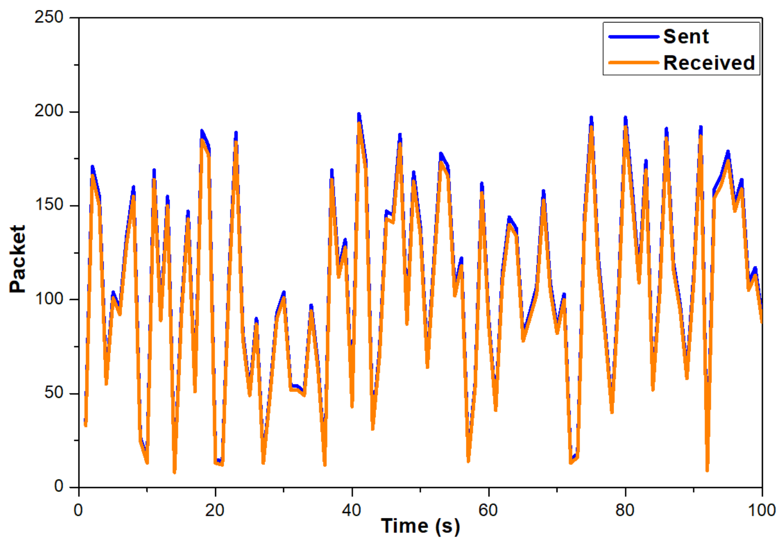


Figure 3.8 – Switch-to-Controller Bandwidth Consumption with mitigation system

switch-controller communication channel, and the space of the switching flow table.

3.5.4 Evaluation & Comparison

Under a DDoS flooding attack, the switch receives many TCP and UDP requests, from random IP sources, which will be forwarded to the controller. These packets may saturate the channel between control and data planes, the couple of SDN controller and switches, as well as the downstream servers.

The proposed mitigation mechanism enables the programmable data plane to handle the incoming packets and forward to the controller only the requests of the successful TCP connections. It discharges the controller-switch path and reduces the involvement of SDN controller. Thus, the SDN network will be more protected from Controller-aimed DDoS attack and flow-table overloading attack. Moreover, it defends the downstream servers from DDoS attacks. Consequently, the proposed solution makes the SDN architecture more scalable and efficient to resist such attacks.

Compared to the security solution proposed in the first contribution 'SDN-based SYN Flooding Defense in Cloud', the current mitigation mechanism:

- does not require or consume a large data plane memory, as it does not use stateful memories (e.g., counters) to store information about incoming packets, such as traffic information DBs extension in the first contribution.
- the high processing power capability of SDN data plane led us exploiting and implementing the SYN flooding mitigation methods in switch, without using a traffic anomaly detection mechanism as used in the first solution.
- protects the SDN controller and switches, the controller-switch path, as well as the downstream servers from DDoS flooding attacks.

Our prototype uses the SYN cookie methods which doesn't require storage of network states in the data plane. This is in contrast to the previous works, such as Avant-guard [82], which use SYN proxy method to protect SDN controller against SYN flood attacks. The SYN proxy requires the information storage (timestamp, sequence number, source IP and port) throughout the TCP connections, which gives rise to a new type of SYN flooding attack named Buffer Saturation Attack.

The implementation of our proposal uses a software-based environment to direct the packet handling process inside the switch, unlike the traditional hardware-based implementation which requires hard and heavy changes to extend and customize the data-plane.

In comparison to some existing solutions, our approach implements simple method functionalities over SDN data plane rather than complex methods, such as machine learning or deep learning which require high memory and processing requirements [107, 108].

In addition to improving resiliency against TCP and UDP Flood attacks, it can also be used to defend against attacks based on HTTP, SMTP and other protocols. This can be done using specific SYN cookie methods, such as HTTP redirect and TCP [7] secure reset.

3.6 Conclusion

In this chapter, we have designed and implemented a lightweight and practical mitigation mechanism to protect SDN architecture against DDoS flooding attack. The proposed approach enabled the programmable data plane to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures. In comparison to the existing solutions, our approach activates the mitigation of DDoS flooding attack at the SDN data plane without any external and dedicated appliance. Consequently, it prevents and reduces the risk of saturation attack in the SDN controller, switches, and the communication channel between them.

With the P4-programmable switch, it was easier to build and implement the security functions offered than the traditional fixed-function OpenFlow switch. Moreover, the simulation results indicate that the proposed mechanism may efficiently tackle the DDoS flood attacks in both SDN architecture and downstream servers.

Now, with a secure and efficient SDN architecture capable of withstanding DDoS attacks, we can move on to designing and implementing a secure SDN-based cloud environment, which is the focus of the next chapter.

Chapter 4

Secure and Efficient SDN-based Cloud Architecture

4.1 Introduction

SDN-based cloud or software-defined cloud networking is a new form of cloud, in which SDN technology allows global view and centralized control of network infrastructure and provides Network as a Service (NaaS) in the cloud computing environment [5], [11], [12], [13]. In SDN-based cloud, cloud computing extends from compute virtualization to storage and network virtualization to get secure and isolated virtual private clouds (VPCs). Cloud providers have already adopted SDN technology in their infrastructure like Google, Vmware, etc.

However, new security issues and particularly new trends of Distributed Deny of service (DDoS) attacks have been introduced over the combination of SDN and cloud computing technologies, as we can refer to the Section. 1.4 and [14], [11] , [15], [16]. DDoS attack may happen when an attacker forces a targeted cloud server to use excessive amounts of finite system resources like network bandwidth, memory, CPU, or disk space, which render services and computing resources unavailable. For instance, the controller saturation attack exhausts the controller and the communication channel between the controller and switches, as well as the space of the switch flow-tables. The good capacities of SDN, such as software-based traffic analysis, centralized control, and dynamic network reconfiguration may greatly improve the detection and mitigation of DDoS attacks in SDN-based cloud environment [16], [17], [18], [11].

Our research work is motivated by this problem, and we ultimately intend to protect SDN-based cloud environment and to make it more resilient against DDoS attacks.

To the best of our knowledge, there are limited research works which address the potential challenges to mitigate DDoS attacks in the SDN-based cloud computing environment. Among which we have presented and analyzed some research works, as shown in Table. 1.2 in Section. 1.5.

In this contribution, we propose an efficient system design that presents the SDN-based Openstack Architecture. The proposed design architecture uses the Openstack opensource cloud platform and the ONOS SDN controller. In this proposal we aim to activate a high-performance and reconfigurable datapath based on the combination of the Programming Protocol-independent Packet Processors (P4) [4], the P4 Runtime API [111], and the Open vSwitch (OvS)[35] software switch.

To protect the SDN-based Openstack platform, we opt to implement the proposed and developed DDoS mitigation mechanism of the previous contribution (see Chapter. 3) that aims to protect the SDN components from DDoS flood attacks, and achieve an efficient networking system. As a result, we get a secure and resilient SDN-based Openstack platform that can resist DDoS flooding attacks.

The present chapter is organized as follows: In Section. 4.2 we present Openstack cloud platform and its services. The Proposed approach is presented in Section. 4.3, in which we describe the SDN-based Openstack architecture and its components, and the Mitigation system used to defend from DDoS attacks.

4.2 Selected technologies

Openstack cloud platform

Openstack is an open source cloud platform widely deployed as IaaS in private and public clouds. Openstack platform consists of a suite of projects managing large pools of compute, storage, and network resources throughout a datacenter. A web-based dashboard is available to manage the presented cloud platform. Openstack is realised in 2010 by Rackspace Hosting and NASA. The platform consists of various components including Nova, Neutron, Keystone, Glance, and Placement.

- **Compute node (Nova)** is the main Openstack component, which controls computing instances. Nova service gives a way to provision compute instances, which is used to host and control cloud computing systems. Compute node runs using the virtualization technologies such as kernel-based VM (KVM), XenServer, Vmware and Hyper-v to create, manage, and remove virtual machines. Nova project also performs a networking service agent that connects instances to virtual networks and provides extension services to instances such as firewalling.

- **Networking service (Neutron)** is an Openstack networking service which provides network connectivity for instances managed by the other Openstack projects like compute service. It allows users to create and manage a virtual networking infrastructure including networks, switches, subnets, and routers. Neutron implements plugins to accommodate different networking equipment and software such as Cisco virtual and physical switches, VMware NSX product, Linux bridging, NEC OpenFlow products, and Open vSwitch. This option provides flexibility to the Openstack architecture and its deployment.

Nova and Neutron services require additional OpenStack services for basic function, including:

- Keystone provides identity and authentication for all OpenStack services.

- Glance provides the compute image repository. All compute instances launch from glance images.

- Placement is responsible for tracking inventory of resources available in a cloud and assisting in choosing which provider of those resources will be used when creating a virtual machine.

4.3 Proposed approach

The purpose of our research work, is the design of an efficient and resilient SDN-based cloud environment that can resist DDoS threats. To this end, we firstly provide an efficient and a secure SDN-based cloud architecture based on the open source Openstack cloud platform, the ONOS SDN controller, and the Open vSwitch software switch. The proposal aims to activate a high-performance and reconfigurable data path based on the integration of the Programming Protocol-independent Packet Processors, the P4Runtime API, and the OvS software switch. Moreover, we use ONOS controller to get visibility and control over the entire network.

Since it offers a reconfigurable data path, the architecture allows administrators to easily activate new network operations such as monitoring, routing, virtualization, firewall, detection, mitigation, etc. In contrast to the traditional OpenFlow and OvS implementation which requires complex design and several lines to add and modify to implement new functions which operate with new header fields.

To defend the SDN-based cloud system against DDoS attacks, we integrate the lightweight and convenient mitigation system that has been provided in the second contribution (Chapter. 3). This latter protect the SDN components including the centralized controller, switches, and southbound interface, as well as the downstream cloud servers.

The goal of our proposed DDoS mitigation mechanism is to activate P4 or reconfigurable data path with smart and advanced functions to prevent and mitigate DDoS flood attacks in the SDN-based cloud environment. To reach this aim, we activate the reconfigurable data path with the classification and mitigation module to analyze and classify the new incoming packets and perform the adaptive countermeasures. In addition we incorporate the control application which will be implemented in the ONOS controller. This application allows the controller to interact with the P4 data path through the P4Runtime interface.

4.3.1 System components and Architecture:

The designed architecture and components of the proposed approach are presented in Figures. 4.1 and 4.2.

CLoud Manager - we select Openstack as cloud manager to host and control cloud resources. Compute resources are managed using Nova service via resource management API. Networking resources are managed using Neutron networking service via northbound API. Neutron communicates with ONOS controller to control the entire physical and virtual network infrastructure. ONOS controller communicates with ovs-vswitchd module via P4 Runtime API. We choose ONOS as the SDN controller because it is the only one that supports P4 switches and P4 Runtime interface, it is also the standard SDN controller developed by the Open Networking Foundation (ONF). Compute servers host VMs using Xenserver hypervisor which

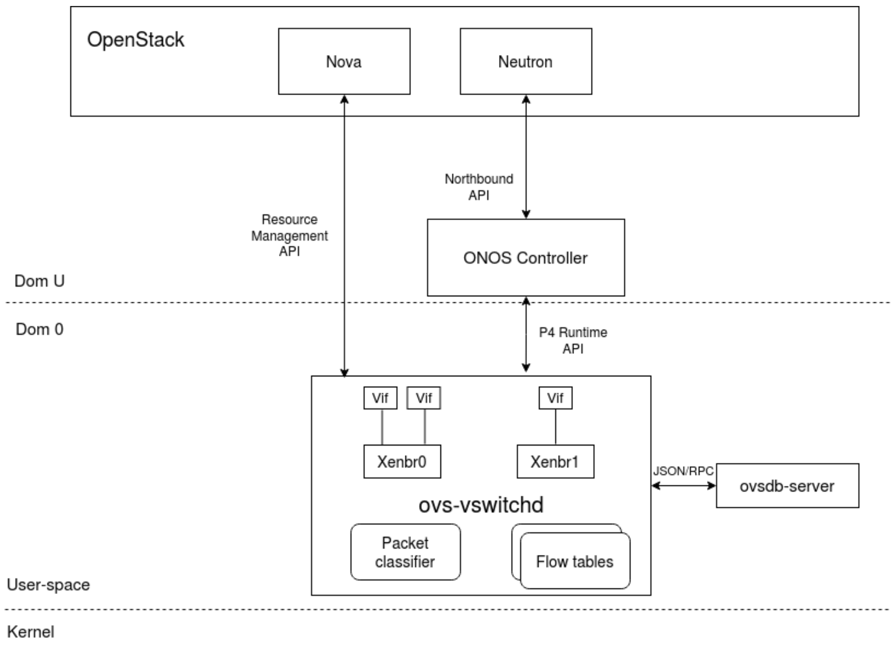


Figure 4.1 – SDN-based Openstack Overall Architecture

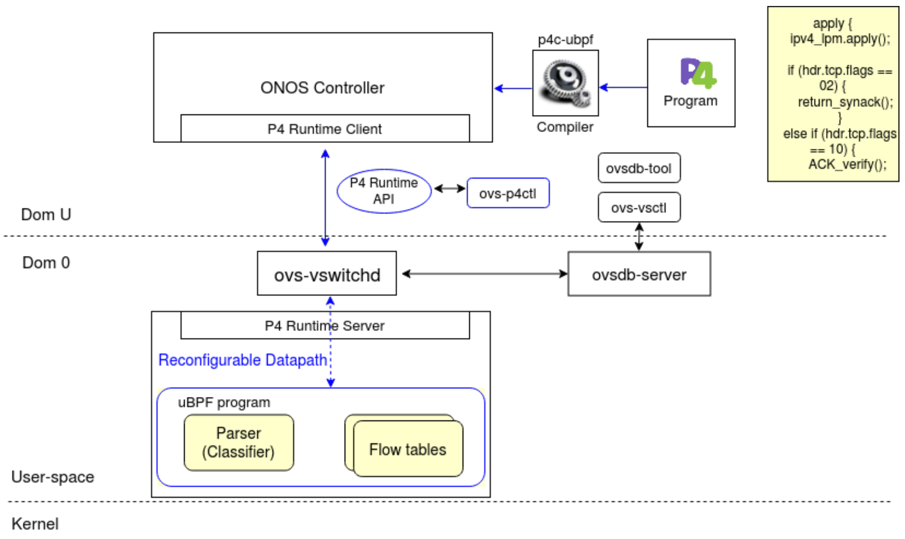
is an efficient virtualization solution. There are two types of domains in Xen-based cloud: Dom 0 and Dom U. Dom 0 is the management domain that belongs to the cloud administrative domain. Dom U is dedicated for administrative purpose to place controller and journal component.

Open vSwitch (OvS) is the widely used software switch in cloud environment. In our established system, OvS is natively implemented in the Dom 0 of XenServer cloud system. The communication between VMs within the same server and those of different servers is controlled by the OvS. In user-space, there are two modules: ovs-switchd and ovsdb-server. The ovs-switchd module is the core component of OvS that supports multiple independent data paths (bridges). The ovsdb-server module holds switch configuration. As shown in Figure. 4.1, the ovs-switchd communicates with the ovsdb-server via the management protocol (e.g., JSON or RPC). The ovs-switchd module communicates with the ONOS controller via P4 Runtime API, and with the kernel module through netlink protocol. In our system, OvS operates entirely in userspace without changing kernel source code or loading kernel modules.

OvS is the software implementation of the OpenFlow switch. OvS can also operate as software implementation of P4-enabled switch. Our implementation is based on the *P4-OvS* project which extends the OvS with support for the P4-enabled datapath and the P4Runtime API to create a high-performance P4 software switch [112]. We use this software implementation to perform our proposed mitigation system in SDN-based Openstack Cloud platform.

The *P4-OvS* project extends OvS with new blocks [112], as shown in Figure. 4.2:

- *Reconfigurable (or P4) datapath* handles the incoming packets using the packet processing pipeline generated from the P4 program, at runtime. The example of supported reconfigurable datapath can be eBPF (Extended Berkeley Packet



```
apply {
ipv4_lpm.apply();

if (hdr.tcp.flags ==
02) {
return_synack();
}
else if (hdr.tcp.flags
== 10) {
ACK_verify();
```

Figure 4.2 – OvS and P4 Architecture

Filter) [113], XDP (eXpress Data Path) or uBPF (userspace BPF) [114].

- *P4 Runtime API* allows communication between the reconfigurable datapath and the SDN controller. In particular, it implements gRPC server with P4Runtime protocol and allows users to control P4 datapath.
- *P4 compiler (p4c-ubpf)* allows to generate datapath-specific binary from the P4 program and P4Info metadata file for P4Runtime API.
- *Ovs-p4ctl* is the management utility tool for the P4 bridge. It manages P4 programs and controls P4 objects (e.g. P4 tables, registers, etc.). The other management tasks (e.g. adding new port) are still implemented by other OvS utility tools (e.g. ovs-vsctl).

In this soft implementation, we use uBPF datapath. uBPF is the userspace re-implementation of in-kernel eBPF VM which provides a user-space execution environment that is extensible at runtime [114].

4.3.2 OvS and P4 Workflow

Figure. 4.3 depicts the programming workflow of P4-OvS. The proposed design architecture enables the administrator to create simple as well as advanced P4 programs (in P4 language) to describe how the incoming packets are processed in the reconfigurable datapath, at runtime. Then the P4 program(s) is compiled using the ubpf compiler (p4c-ubpf). This latter generates uBPF bytecode (datapath code) from the P4 program and the P4Info metadata file to be used by the control plane as a contract describing the datapath implementation. The ovs-vsctl tool allows the administrator to create P4 bridge with uBPF datapath and will inject the target binary (uBPF bytecode) to the uBPF datapath, so that packets can be handled by the configured packet processing pipeline. Then, the administrator can simply manage and configure P4 bridge(s) via ovs-p4ctl tool.

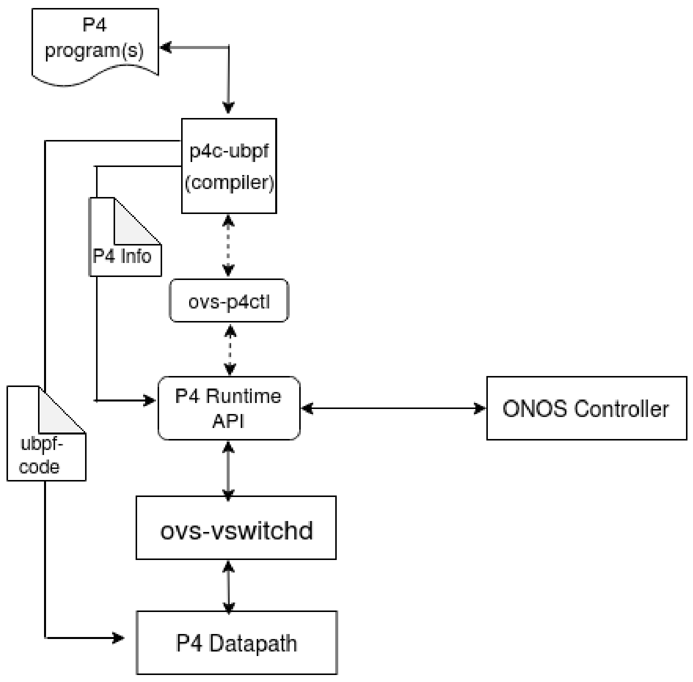


Figure 4.3 – OvS and P4 Workflow

4.3.3 DDoS mitigation mechanism

Classification and Mitigation application allows the reconfigurable data path (DP) to analyze and classify the new incoming packets, the benign packets from the SYN flood attacks. In first, DP checks if the packet exists in the flow table if so, the packet will be immediately forwarded to the destination server. Otherwise, it responds to the SYN request client by a SYN-ACK packet with a pre-generated cookie (see Subsection. 2.3.3). The cookie or the Initial Sequence Number (ISN) is created by hashing details about the initial SYN packet and its TCP options.

When the DP receives an ACK packet, it checks the validation of the TCP sequence number, see pre-generated cookie in Subsection. 2.3.3). If the TCP connection is validated the DP transmits the ACK packet to the ONOS controller to write the required flow-entry. At the same time, the DP sends back a TCP-reset packet (i.e., with source IP of the original server) to the client in order to enable him to re-establish the connection directly with the server. The ONOS controller is responsible for controlling the P4 software switches.

Control application is developed in python language, it will be implemented in the ONOS controller. This application allows the controller to interact with the P4 data path using the P4 Runtime interface. The application connects the controller to the switch(s), installs the classification and mitigation P4-programs on the switch(s), populates the defined flow-tables with flow rules, and describes how the ONOS controller handles the incoming messages (packet-in).

When the controller receives a packet-in message it learns from the details of the initial SYN packet to create the requested flow-rule. It extracts the addresses of IP source, IP destination, Mac source, and Mac destination, and the ports of source and destination. It checks if the new flow has already a flow-rule in the memory of the controller, if it exists it writes it on the switch, if not it creates a new flow rule for the new flow and writes it on switch flow-table. Each flow rule is installed with a timeout value, both hard and idle timeouts, to avert the growth of rules in the switch flow-tables.

4.4 Conclusion

The growing adoption of cloud and SDN technologies have highlighted the requirement to analyze and evaluate the benefits and vulnerabilities of these technologies. While DDoS attacks remain a top threat that is growing in size and frequency of reported incidents, SDN-based cloud opens the door for yet new vulnerabilities to this type of attacks.

In this contribution, we have proposed an efficient system design that presents the SDN-based Openstack Architecture. In the proposed design, we have activated a high-performance and reconfigurable datapath based on the combination of the Programming Protocol-independent Packet Processors (P4), the P4 Runtime API, and the Open vSwitch (OvS) software switch.

To protect the SDN-based Openstack Architecture, we have used the lightweight and practical mitigation mechanism to defend the SDN architecture and cloud services against DDoS flooding attacks. The proposed approach enables the datapath to analyze the new incoming packets, classify the benign requests from the SYN flood attacks, and perform the adaptive countermeasures.

Conclusion and Future Works

High flexibility and scalability along with reduced infrastructure cost let cloud technology widely used in the public and businesses. This growing adoption has highlighted the need to protect the computing, storage and network resources of the cloud environment. Since everything in the cloud is defined as service provided on-demand, availability is the crucial security requirement in cloud environment. Deny of Service is the main attack disrupting the availability of cloud services. The essential features of SDN technology, including software-based traffic analysis, centralized control over the network, and dynamic network reconfiguration, etc. may greatly improve the protection against DDoS attacks in the cloud environment. In this end, we have made various observable contributions.

The first contribution aims to design a defensive mechanism which enables the SDN data plane to prevent and mitigate DDOS flooding attacks in cloud environment. The proposed solution extends the data plane with detection and mitigation techniques. The second proposal aims to develop a lightweight and practical mitigation mechanism to protect SDN architecture against DDoS flooding threats and ensure a secure and resilient SDN-based networking environment. The third contribution concerns the design of an efficient and a secure SDN-based Openstack System Architecture. The proposal aims to enable a high-performance and reconfigurable datapath based on the combination of the Programming Protocol-independent Packet Processors (P4), the P4 Runtime API, and the Open vSwitch software switch.

To prove the feasibility of our proposals, we have implemented our solutions using the P4 programming language [4] and bmv2 (behavioral-model) software switch in Mininet emulator. The simulation results are satisfactory compared to existing methods.

The proposed mechanisms are useful to defend SDN-based cloud environment from DDoS threats. However, there are many more diverse network attacks, and we want to protect the cloud network against these attacks.

To make our solutions more scalable, we aim to investigate existing load balancing methods for managing switches that enable security functions in large cloud data centers.

In future works, we aim to harness the benefits of SDN, cloud, and other technologies such as Network Function Virtualization (NFV) to provide network sharing and simplify management operations in current application areas, such as cyber-physical systems and the Internet of Things (IoT).

Publications

Journal publications

[1] Safaa Mahrach and Abdelkrim Haqiq. "DDoS Flooding Attack Mitigation in Software Defined Networks". International Journal of Advanced Computer Science and Applications, Vol. 11, pp. 693-700, 2020.

[2] Safaa Mahrach, Iman El mir, Abdelkrim Haqiq, and Dijiang Huang. "SDN-based SYN Flooding Defense in Cloud". Journal of Information Assurance and Security, Vol. 13, pp. 30-39, 2018.

Chapter Book

[3] Safaa Mahrach and Abdelkrim Haqiq. "DDoS defense in SDN-based Cyber-Physical Cloud". Cybersecurity and Privacy in Cyber Physical Systems, pp. 133-158, 2019.

Communications in conference proceedings

[4] Safaa Mahrach and Abdelkrim Haqiq. "DDoS Attack and Defense in SDN-based Cloud". In the Proceedings of the International Symposium on Ubiquitous Networking (UNet'21), held online, 19-21 May, 2021. Springer LNCS, 2021.

[5] Safaa Mahrach, Oussama Mjihil, and Abdelkrim Haqiq. "Scalable and Dynamic Network Intrusion Detection and Prevention System". In the Proceedings of the International Conference on Innovations in Bio-Inspired Computing and Applications (IBICA 2017), held in Marrakesh, Morocco, 11-13 December, 2017. Published on the book of Advances in Intelligent Systems and Computing, Vol. 735, pp. 318-328, Springer, 2017.

Bibliography

[1] The open networking foundation. https://www.opennetworking.org/, accessed on 20th June 2017. x, 13

[2] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. Openflow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008. x, 14

[3] Frederik Hauser, Marco Häberle, Daniel Merling, Steffen Lindner, Vladimir Gurevich, Florian Zeiger, Reinhard Frank, and Michael Menth. A survey on data plane programming with p4: Fundamentals, advances, and applied research. *arXiv preprint arXiv:2101.10632*, 2021. x, 16

[4] The P4 Language Specification. https://p4.org/p4-spec/p4-14/v1.0.4/tex/p4.pdf, accessed on 30th May 2017. x, xi, 5, 13, 17, 18, 30, 33, 36, 45, 48, 57, 63, 73, 79, 91, 92, 93, 95

[5] Jungmin Son and Rajkumar Buyya. A taxonomy of software-defined networking (sdn)-enabled cloud computing. *ACM Computing Surveys (CSUR)*, 51(3):59, 2018. x, 1, 6, 15, 16, 19, 20, 26, 27, 72

[6] Fang-Yie Leu and Zhi-Yang Li. Detecting dos and ddos attacks by using an intrusion detection and remote prevention system. In *Information Assurance and Security, 2009. IAS'09. Fifth International Conference on*, volume 2, pages 251–254. IEEE, 2009. x, 38, 39

[7] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. Network anti-spoofing with sdn data plane. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017. x, 36, 40, 41, 56, 58, 62, 70

[8] Safaa Mahrach and Abdelkrim Haqiq. Ddos defense in sdn-based cyber-physical cloud. *Cybersecurity and Privacy in Cyber Physical Systems*, page 133, 2019. 1, 30, 55

[9] Shruti Wadhwa and Vipul Mandhar. Survey on ddos and edos attack in cloud environment. *Evolving Technologies for Computing, Communication and Smart World*, pages 317–332, 2021. 1

[10] Wu Zhijun, Li Wenjing, Liu Liang, and Yue Meng. Low-rate dos attacks, detection, defense, and challenges: a survey. *IEEE Access*, 8:43920–43943, 2020. 1

[11] Qiao Yan, F Richard Yu, Qingxiang Gong, and Jianqiang Li. Software-defined networking (sdn) and distributed denial of service (ddos) attacks in cloud computing environments: A survey, some research issues, and challenges. *IEEE Communications Surveys & Tutorials*, 18(1):602–622, 2016. 1, 2, 6, 19, 26, 27, 29, 43, 72

[12] Jungmin Son, Amir Vahid Dastjerdi, Rodrigo N Calheiros, and Rajkumar Buyya. Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers. *IEEE Transactions on Sustainable Computing*, 2(2):76–89, 2017. 1, 6, 16, 19, 72

[13] Richard Cziva, Simon Jouët, David Stapleton, Fung Po Tso, and Dimitrios P Pezaros. Sdn-based virtual machine management for cloud data centers. *IEEE Transactions on Network and Service Management*, 13(2):212–225, 2016. 1, 6, 15, 19, 72

[14] Kriti Bhushan and BB Gupta. Security challenges in cloud computing: state-of-art. *International Journal of Big Data Intelligence*, 4(2):81–107, 2017. 1, 6, 72

[15] Gaurav Somani, Manoj Singh Gaur, Dheeraj Sanghi, Mauro Conti, and Rajkumar Buyya. Ddos attacks in cloud computing: Issues, taxonomy, and future directions. *Computer Communications*, 107:30–48, 2017. 1, 6, 29, 43, 72

[16] Kriti Bhushan and BB Gupta. Distributed denial of service (ddos) attack mitigation in software defined network (sdn)-based cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*, pages 1–13, 2018. 1, 2, 6, 26, 27, 30, 31, 52, 72

[17] Hubert D'Cruze, Ping Wang, Raed Omar Sbeit, and Andrew Ray. A software-defined networking (sdn) approach to mitigating ddos attacks. In *Information Technology-New Generations*, pages 141–145. Springer, 2018. 1, 6, 30, 72

[18] Bing Wang, Yao Zheng, Wenjing Lou, and Y Thomas Hou. Ddos attack protection in the era of cloud computing and software-defined networking. *Computer Networks*, 81:308–319, 2015. 1, 2, 6, 30, 31, 43, 52, 55, 72

[19] Narmeen Zakaria Bawany, Jawwad A Shamsi, and Khaled Salah. Ddos attack detection and mitigation using sdn: methods, practices, and solutions. *Arabian Journal for Science and Engineering*, 42(2):425–441, 2017. 2, 6, 30, 31, 33, 52

[20] Q Yan, Q Gong, and FR Yu. Effective software-defined networking controller scheduling method to mitigate ddos attacks. *Electronics Letters*, 53(7):469–471, 2017. 2, 6, 30, 31, 33, 52, 55

[21] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014. 2, 33, 37

[22] Shuyong Zhu, Jun Bi, Chen Sun, Chenhui Wu, and Hongxin Hu. Sdpa: Enhancing stateful forwarding for software-defined networking. In *Network Protocols (ICNP), 2015 IEEE 23rd International Conference on*, pages 323–333. IEEE, 2015. 2, 33, 37

[23] Mina Tahmasbi Arashloo, Yaron Koral, Michael Greenberg, Jennifer Rexford, and David Walker. Snap: Stateful network-wide abstractions for packet processing. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 29–43. ACM, 2016. 2, 33, 34, 37

[24] Anirudh Sivaraman, Alvin Cheung, Mihai Budiu, Changhoon Kim, Mohammad Alizadeh, Hari Balakrishnan, George Varghese, Nick McKeown, and Steve Licking. Packet transactions: High-level programming for line-rate switches. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 15–28. ACM, 2016. 2, 33, 34

[25] Cloud Economics: Making the Business Case for Cloud. https://assets.kpmg.com/content/dam/kpmg/pdf/2015/11/cloud-economics.pdf, accessed on 25th January 2016. 6

[26] Sara Farahmandian and Doan B Hoang. Security for software-defined (cloud, sdn and nfv) infrastructures–issues and challenges. In *Eight International Conference on Network and Communications Security*, 2016. 6, 26, 27

[27] Wikipedia 'Cloud computing definition'. https://en.wikipedia.org/wiki/Cloud computing, accessed on 20th June 2020. 7

[28] The NIST Definition of Cloud Computing. https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication 800-145.pdf, accessed on 22th January 2018. 7

[29] The BSI Definition of Cloud Computing. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Publications/CloudComputing/Se
RecommendationsCloudComputingProviders.pdf, accessed on 14th January 2020. 7

[30] Yosr Jarraya, Taous Madi, and Mourad Debbabi. A survey and a layered taxonomy of software-defined networking. *IEEE communications surveys & tutorials*, 16(4):1955–1980, 2014. 13

[31] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, J Hadi Salim, David Meyer, and Odysseas Koufopavlou. Software-defined networking (sdn): Layers and architecture terminology. Technical report, 2015. 13

[32] Diego Kreutz, Fernando MV Ramos, Paulo Esteves Verissimo, Christian Esteve Rothenberg, Siamak Azodolmolky, and Steve Uhlig. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76, 2015. 13

[33] OpenFlow Switch Specification. 1.4 (2019) https://www. opennetworking. org/wp-content/uploads/2014/10/openflow-spec-v1. 4.0. pdf, 2019. 13

[34] Network Configuration Protocol (NETCONF). https://tools.ietf.org/html/rfc6241, accessed on 10th February 2019. 13

[35] Open vSwitch. https://www.openvswitch.org/, accessed on 30th October 2020. 13, 73

[36] Pica8. https://www.pica8.com/, accessed on 28th October 2019. 13

[37] Sandeep Pisharody, Janakarajan Natarajan, Ankur Chowdhary, Abdullah Alshalan, and Dijiang Huang. Brew: A security policy analysis framework for distributed sdn-based cloud environments. *IEEE Transactions on Dependable and Secure Computing*, 2017. 15

[38] Xue Leng, Kaiyu Hou, Yan Chen, Kai Bu, Libin Song, and You Li. A lightweight policy enforcement system for resource protection and management in the sdn-based cloud. *Computer Networks*, 161:68–81, 2019. 16, 19

[39] Programming The Network Data Plane: What, How, and Why? https://conferences.sigcomm.org/events/apnet2017/slides/chang.pdf, accessed on 10th October 2020. 16

[40] Chaowei Yang, Qunying Huang, Zhenlong Li, Kai Liu, and Fei Hu. Big data and cloud computing: innovation opportunities and challenges. *International Journal of Digital Earth*, 10(1):13–53, 2017. 18

[41] Mohamed Almorsy, John Grundy, and Ingo Müller. An analysis of the cloud computing security problem. *arXiv preprint arXiv:1609.01107*, 2016. 18, 29

[42] Saurabh Singh, Young-Sik Jeong, and Jong Hyuk Park. A survey on cloud computing security: Issues, threats, and solutions. *Journal of Network and Computer Applications*, 75:200–222, 2016. 18

[43] John W Rittinghouse and James F Ransome. *Cloud computing: implementation, management, and security*. CRC press, 2016. 18

[44] Blesson Varghese and Rajkumar Buyya. Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems*, 79:849–861, 2018. 18

[45] Haijun Zhang, Na Liu, Xiaoli Chu, Keping Long, Abdol-Hamid Aghvami, and Victor CM Leung. Network slicing based 5g and future mobile networks: mobility, resource management, and challenges. *IEEE Communications Magazine*, 55(8):138–145, 2017. 18

[46] Manuel Díaz, Cristian Martín, and Bartolomé Rubio. State-of-the-art, challenges, and open issues in the integration of internet of things and cloud computing. *Journal of Network and Computer Applications*, 67:99–117, 2016. 19

[47] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017. 19

[48] Ren-Hung Hwang, Huei-Ping Tseng, and Yu-Chi Tang. Design of sdn-enabled cloud data center. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, pages 950–957. IEEE, 2015. 19

[49] Yaser Jararweh, Mahmoud Al-Ayyoub, Elhadj Benkhelifa, Mladen Vouk, Andy Rindos, et al. Software defined cloud: Survey, system and evaluation. *Future Generation Computer Systems*, 58:56–74, 2016. 19

[50] Rajkumar Buyya, Rodrigo N Calheiros, Jungmin Son, Amir Vahid Dastjerdi, and Young Yoon. Software-defined cloud computing: Architectural elements and open challenges. In *2014 International conference on advances in computing, communications and informatics (ICACCI)*, pages 1–12. IEEE, 2014. 19

[51] Openstack. https://www.openstack.org/, accessed on 10th August 2020. 19

[52] The Cloud Security Alliance. https://cloudsecurityalliance.org/, accessed on 3th November 2018. 25

[53] Treacherous 12: Top Threats to Cloud Computing (2016). https://downloads.cloudsecurityalliance.org/assets/research/top-threats/Treacherous-12_Cloud-Computing_Top-Threats.pd, accessed on 3th November 2018. 25

[54] Heng Zhang, Zhiping Cai, Qiang Liu, Qingjun Xiao, Yangyang Li, and Chak Fone Cheang. A survey on security-aware measurement in sdn. *Security and Communication Networks*, 2018, 2018. 26

[55] Sandra Scott-Hayward, Gemma O'Callaghan, and Sakir Sezer. Sdn security: A survey. In *Future Networks and Services (SDN4FNS), 2013 IEEE SDN For*, pages 1–7. IEEE, 2013. 26

[56] Danda B Rawat and Swetha R Reddy. Software defined networking architecture, security and energy efficiency: A survey. *environment*, 3(5):6, 2017. 26

[57] Abdallah Mustafa Abdelrahman, Joel JPC Rodrigues, Mukhtar ME Mahmoud, Kashif Saleem, Ashok Kumar Das, Valery Korotaev, and Sergei A Kozlov. Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions. *International Journal of Communication Systems*, page e4706. 27, 28

[58] Shilpa P Khedkar and R AroulCanessane. Sdn enabled cloud, iot and dcns: A comprehensive survey. In *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, pages 1–5. IEEE, 2019. 27

[59] R Kalaiprasath, R Elankavi, Dr R Udayakumar, et al. Cloud. security and compliance-a semantic approach in end to end security. *International Journal Of Mechanical Engineering And Technology (Ijmet)*, 8(5), 2017. 29

[60] Rup Kumar Deka, Dhruba Kumar Bhattacharyya, and Jugal Kumar Kalita. Ddos attacks: Tools, mitigation approaches, and probable impact on private cloud environment. *arXiv preprint arXiv:1710.08628*, 2017. 29

[61] The connection tracking system. http://docs.openvswitch.org/en/latest/tutorials/ovs-conntrack/, accessed on 3th September 2020. 30

[62] Giuseppe Bianchi, Marco Bonola, Antonio Capone, and Carmelo Cascone. Openstate: programming platform-independent stateful openflow applications inside the switch. *ACM SIGCOMM Computer Communication Review*, 44(2):44–51, 2014. 30

[63] Tooska Dargahi, Alberto Caponi, Moreno Ambrosin, Giuseppe Bianchi, and Mauro Conti. A survey on the security of stateful sdn data planes. *IEEE Communications Surveys & Tutorials*, 2017. 33, 37

[64] Arash Shaghaghi, Mohamed Ali Kaafar, Rajkumar Buyya, and Sanjay Jha. Software-defined network (sdn) data plane security: Issues, solutions and future directions. *arXiv preprint arXiv:1804.00262*, 2018. 33

[65] Celio Trois, Marcos D Del Fabro, Luis CE de Bona, and Magnos Martinello. A survey on sdn programming languages: Toward a taxonomy. *IEEE Communications Surveys & Tutorials*, 18(4):2687–2712, 2016. 33

[66] Masoud Moshref, Apoorv Bhargava, Adhip Gupta, Minlan Yu, and Ramesh Govindan. Flow-level state transition as a new switch primitive for sdn. In *Proceedings of the third workshop on Hot topics in software defined networking*, pages 61–66. ACM, 2014. 33, 37

[67] Giuseppe Bianchi, Marco Bonola, Antonio Capone, and Carmelo Cascone. Openstate: programming platform-independent stateful openflow applications inside the switch. *ACM SIGCOMM Computer Communication Review*, 44(2):44–51, 2014. 33, 37

[68] Péter Vörös and Attila Kiss. Security middleware programming using p4. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*, pages 277–287. Springer, 2016. 34

[69] Yehuda Afek, Anat Bremler-Barr, and Lior Shafir. Network anti-spoofing with sdn data plane. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017. 34, 52

[70] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007. 35

[71] V Balamurugan and R Saravanan. Enhanced intrusion detection and prevention system on cloud environment using hybrid classification and ots generation. *Cluster Computing*, pages 1–13, 2017. 35

[72] Zakira Inayat, Abdullah Gani, Nor Badrul Anuar, Shahid Anwar, and Muhammad Khurram Khan. Cloud-based intrusion detection and response system: open research issues, and solutions. *Arabian Journal for Science and Engineering*, 42(2):399–423, 2017. 35

[73] Tianyi Xing, Dijiang Huang, Le Xu, Chun-Jen Chung, and Pankaj Khatkar. Snortflow: A openflow-based intrusion prevention system in cloud environment. In *Research and Educational Experiment Workshop (GREE), 2013 Second GENI*, pages 89–92. IEEE, 2013. 35

[74] Tianyi Xing, Zhengyang Xiong, Dijiang Huang, and Deep Medhi. Sdnips: Enabling software-defined networking based intrusion prevention system in clouds. In *Network and Service Management (CNSM), 2014 10th International Conference on*, pages 308–311. IEEE, 2014. 35, 36

[75] Yaping Chi, Tingting Jiang, Xiao Li, and Cong Gao. Design and implementation of cloud platform intrusion prevention system based on sdn. In *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*, pages 847–852. IEEE, 2017. 35

[76] Shivam Tiwari, Vanshika Pandita, Samarth Sharma, Vishal Dhande, and Shailesh Bendale. Survey on sdn based network intrusion detection system using machine learning framework, 2019. 35

[77] Daeyoung Hyun, Jinyoug Kim, Dongjin Hong, and Jaehoon Paul Jeong. Sdn-based network security functions for effective ddos attack mitigation. In *2017 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 834–839. IEEE, 2017. 35

[78] Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 413–424. ACM, 2013. 36, 52

[79] John Sonchack, Jonathan M Smith, Adam J Aviv, and Eric Keller. Enabling practical software-defined networking security applications with ofx. In *NDSS*, volume 16, pages 1–15, 2016. 36, 37

[80] Iman El Mir, Ankur Chowdhary, Dijiang Huang, Sandeep Pisharody, Dong Seong Kim, and Abdelkrim Haqiq. Software defined stochastic model for moving target defense. In *International Afro-European Conference for Industrial Advancement*, pages 188–197. Springer, 2016. 36

[81] Changhoon Yoon, Taejune Park, Seungsoo Lee, Heedo Kang, Seungwon Shin, and Zonghua Zhang. Enabling security functions with sdn: A feasibility study. *Computer Networks*, 85:19–35, 2015. 36

[82] Seungwon Shin, Vinod Yegneswaran, Phillip Porras, and Guofei Gu. Avant-guard: Scalable and vigilant switch flow management in software-defined networks. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 413–424, 2013. 37, 56, 70

[83] Moreno Ambrosin, Mauro Conti, Fabio De Gaspari, and Radha Poovendran. Lineswitch: Efficiently managing switch flow in software-defined networking while effectively tackling dos attacks. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, pages 639–644, 2015. 37, 56

[84] Vasilios A Siris and Fotini Papagalou. Application of anomaly detection algorithms for detecting syn flooding attacks. *Computer communications*, 29(9):1433–1442, 2006. 38

[85] Nenekazi NP Mkuzangwe, Andre McDonald, and Fulufhelo V Nelwamondo. Implementation of anomaly detection algorithms for detecting transmission control protocol synchronized flooding attacks. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2015 12th International Conference on*, pages 2137–2141. IEEE, 2015. 38

[86] Pierre Granjon. The cusum algorithm-a small review. 2013. 38

[87] Haining Wang, Danlu Zhang, and Kang G Shin. Detecting syn flooding attacks. In *Proceedings. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1530–1539. IEEE, 2002. 38

[88] Stephen M Fontes, John R Hind, Thomas Narten, and Marcia L Stockton. Blended syn cookies, June 6 2006. US Patent 7,058,718. 39

[89] Dan Touitou, Guy Pazi, Yehiel Shtein, and Rephael Tzadikario. Using tcp to authenticate ip source addresses, July 12 2011. US Patent 7,979,694. 40

[90] Dan Touitou and Rafi Zadikario. Upper-level protocol authentication, May 19 2009. US Patent 7,536,552. 40

[91] RFC 793 (TCP). https://tools.ietf.org/html/rfc793, accessed on 16th June 2018. 41, 44

[92] William Allen Simpson. Tcp cookie transactions (tcpct). 2011. 42

[93] Juan Jose Echevarria, Pablo Garaizar, and Jon Legarda. An experimental study on the applicability of syn cookies to networked constrained devices. *Software: Practice and Experience*. 42, 60

[94] Kevin Thompson, Gregory J Miller, and Rick Wilder. Wide-area internet traffic patterns and characteristics. *IEEE network*, 11(6):10–23, 1997. 45

[95] Hellen Maziku, Sachin Shetty, and David M Nicol. Security risk assessment for sdn-enabled smart grids. *Computer Communications*, 133:1–11, 2019. 54

[96] Safaa Mahrach, Iman El Mir, Abdelkrim Haqiq, and Dijiang Huang. Sdn-based syn flooding defense in cloud. *Journal of Information Assurance and Security*, 13(1), 2018. 54, 56

[97] Carlos Gonzalez, Salim Mahamat Charfadine, Olivier Flauzac, and Florent Nolot. Sdn-based security framework for the iot in distributed grid. In *2016 international multidisciplinary conference on computer and energy science (SpliTech)*, pages 1–5. IEEE, 2016. 54

[98] DDoS Attacks in Q1 2019 report. https://securelist.com/ddos-report-q1-2019/90792/, accessed on 10th December 2019. 54

[99] BEHAVIORAL MODEL (bmv2). https://github.com/p4lang/behavioral-model, accessed on 19th May 2017. 55

[100] Mininet emulator. http://mininet.org/, accessed on 10th Mars 2017. 55

[101] Rochak Swami, Mayank Dave, and Virender Ranga. Software-defined networking-based ddos defense mechanisms. *ACM Computing Surveys (CSUR)*, 52(2):1–36, 2019. 56

[102] Celyn Birkinshaw, Elpida Rouka, and Vassilios G Vassilakis. Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks. *Journal of Network and Computer Applications*, 136:71–85, 2019. 56

[103] Celyn Birkinshaw, Elpida Rouka, and Vassilios G Vassilakis. Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks. *Journal of Network and Computer Applications*, 136:71–85, 2019. 56

[104] Safaa Mahrach, Oussama Mjihil, and Abdelkrim Haqiq. Scalable and dynamic network intrusion detection and prevention system. In *International Conference on Innovations in Bio-Inspired Computing and Applications*, pages 318–328. Springer, 2017. 56

[105] Seyed Mohammad Mousavi and Marc St-Hilaire. Early detection of ddos attacks against software defined network controllers. *Journal of Network and Systems Management*, 26(3):573–591, 2018. 56

[106] Ngoc Thinh Tran, Tan Long Le, and Minh Anh Tuan Tran. Odl-antiflood: A comprehensive solution for securing opendaylight controller. In *2018 International Conference on Advanced Computing and Applications (ACOMP)*, pages 14–21. IEEE, 2018. 56

[107] Tuan Anh Tang, Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, Mounir Ghogho, and Fadi El Moussa. Deepids: Deep learning approach for intrusion detection in software defined networking. *Electronics*, 9(9):1533, 2020. 56, 70

[108] Chuanhuang Li, Yan Wu, Xiaoyong Yuan, Zhengjun Sun, Weiming Wang, Xiaolin Li, and Liang Gong. Detection and defense of ddos attack–based on deep learning in openflow-based sdn. *International Journal of Communication Systems*, 31(5):e3497, 2018. 56, 70

[109] Protecting against spoofed DNS messages. https://patents.google.com/patent/US20030070096. 57

[110] Open Network Operating System (ONOS). https://opennetworking.org/onos/, accessed on 10th June 2020. 58

[111] P4Runtime. https://p4.org/api/p4-runtime-putting-the-control-plane-in-charge-of-the-forwarding-plane.html, accessed on 10th June 2020. 58, 73

[112] P4-OvS - Bringing the power of P4 to OvS! https://github.com/osinstom/P4-OvS, accessed on 1th July 2020. 75

[113] eBPF. https://ebpf.io/, accessed on 3th June 2020. 76

[114] uBPF. https://github.com/iovisor/ubpf, accessed on 10th July 2020. 76

Annex A

A P4-16 Language specification

A.1 Overview

Programming Protocol-independent Packet Processors (P4) is a language for expressing how packets are processed by the data plane of a programmable forwarding element such as a hardware or software switch, network interface card, router, or network appliance [4]. While P4 was initially designed for programming switches, its scope has been broadened to cover a large variety of devices. Many targets(i.e., network devices) implement both a control plane and a data plane. P4 is designed to specify only the data plane functionality of the target. P4 programs also partially define the interface by which the control plane and the data-plane communicate, but P4 cannot be used to describe the control plane functionality of the target.

As a concrete example of a target, Fig. 6.5 illustrates the difference between a traditional fixed-function switch and a P4-programmable switch. In a traditional switch the manufacturer defines the data-plane functionality. The control-plane controls the data plane by managing entries in tables (e.g.routing tables), configuring specialized objects (e.g. meters), and by processing control-packets (e.g.routing protocol packets) or asynchronous events, such as link state changes or learning notifications. A P4-programmable switch differs from a traditional switch in two essential ways:

- The data plane functionality is not fixed in advance but is defined by a P4 program. The data plane is configured at initialization time to implement the functionality described by the P4 program(shown by the long red arrow) and has no built-in knowledge of existing network protocols.

- The control plane communicates with the data plane using the same channels as in a fixed-function device, but the set of tables and other objects in the data plane are no longer fixed, since they are defined by a P4 program. The P4 compiler generates the API that the control plane uses to communicate with the data plane.

Hence, P4 can be said to be protocol independent, but it enables programmers to express a rich set of protocols and other data plane behaviors. The core abstractions provided by the P4 language are:

- Header types describe the format (the set of fields and their sizes) of each header within a packet.

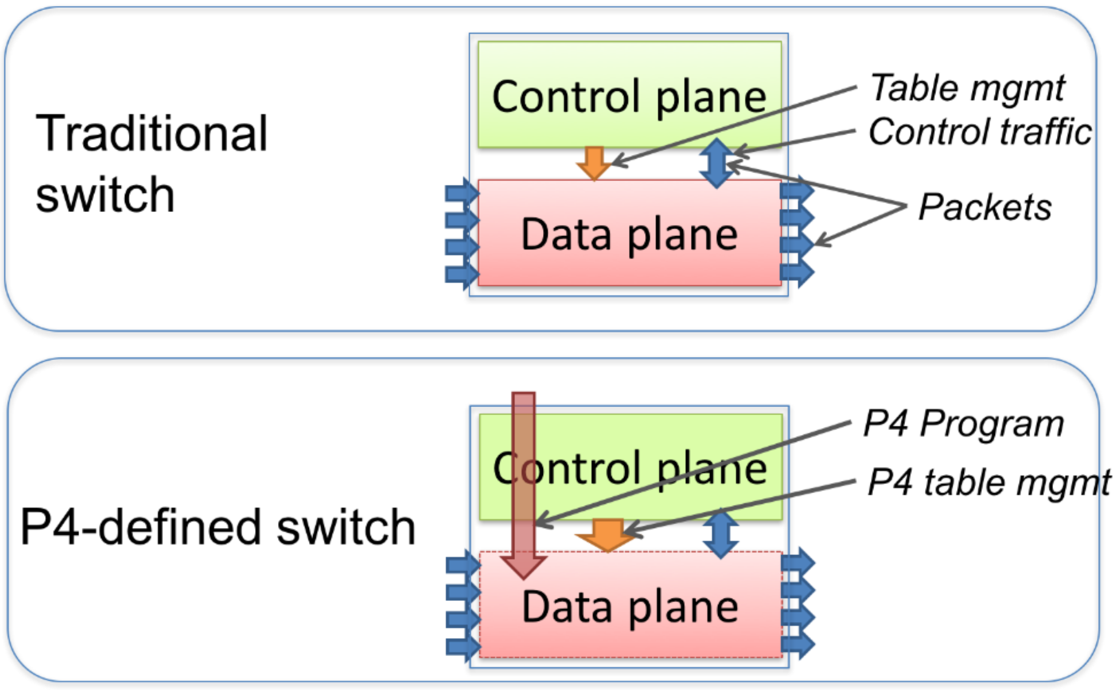


Figure 6.4 – Traditional switches vs. programmable switches [4].

- Parsers describe the permitted sequences of headers within received packets, how to identify those header sequences, and the headers and fields to extract from packets.
- Tables associate user-defined keys with actions. P4 tables generalize traditional switch tables; they can be used to implement routing tables, flow lookup tables, access-control lists, and other user-defined table types, including complex multi-variable decisions.
- Actions are code fragments that describe how packet header fields and metadata are manipulated. Actions can include data, which is supplied by the control-plane at runtime.
- Match-action units perform the following sequence of operations:
  - Construct lookup keys from packet fields or computed metadata,
  - Perform table lookup using the constructed key, choosing an action (including the associated data) to execute,
  - Finally, execute the selected action.
- Control flow expresses an imperative program that describes packet-processing on a target, including the data-dependent sequence of match-action unit invocations. Deparsing (packet reassembly) can also be performed using a control flow.

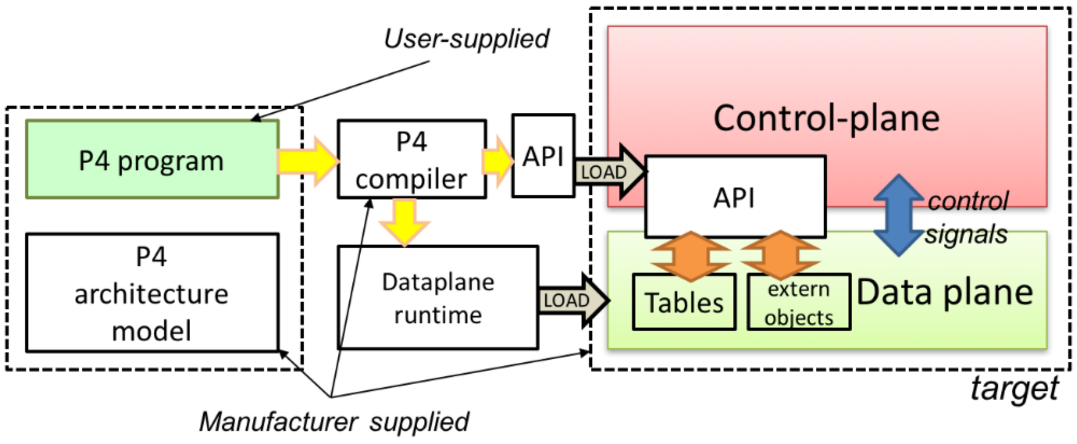


Figure 6.5 – Programming a target with P4 [4].

- Extern objects are architecture-specific constructs that can be manipulated by P4 programs through well-defined APIs, but whose internal behavior is hard-wired (e.g., checksum units) and hence not programmable using P4.
- User-defined metadata: user-defined data structures associated with each packet.
- Intrinsic metadata: metadata provided by the architecture associated with each packet—e.g., the input port where a packet has been received

Fig. 6.5 shows a typical tool workflow when programming the data plane of a target using P4.Target manufacturers provide the hardware or software implementation framework, an architecture definition, and a P4 compiler for that target. P4 programmers write programs for a specific architecture, which defines a set of P4-programmable components on the target as well as their external data plane interfaces.Compiling a set of P4 programs produces two artifacts:

- a data plane configuration that implements the forwarding logic described in the input program
- an API for managing the state of the data plane objects from the control plane

P4 is a domain-specific language that is designed to be implementable on a large variety of targets including programmable network interface cards, FPGAs, software switches, and hardware ASICs. As such, the language is restricted to constructs that can be efficiently implemented on all of these plat-forms.Assuming a fixed cost for table lookup operations and interactions with extern objects, all P4 pro-grams (i.e., parsers and controls) execute a constant number of operations for each byte of an input packet received and analyzed. Although parsers may contain loops, provided some header is extracted on each cycle, the packet itself provides a bound on the total execution of the parser. In other words,under these assumptions, the computational complexity of a P4 program is linear in the total size of all headers, and never depends on the size of the state accumulated while processing data (e.g., the number of flows, or the total number of packets processed). These guarantees are necessary (but not sufficient) for enabling fast packet processing across a variety of targets.

P4 conformance of a target is defined as follows: if a specific target T supports only a subset of the P4 programming language, say P4T, programs written in P4T executed on the target should provide the exact same behavior as is described in this document. Note that P4 conformant targets can provide arbitrary P4 language extensions and **extern** elements.

A.2 Benefits of P4

Compared to state-of-the-art packet-processing systems (e.g., based on writing microcode on top of custom hardware), P4 provides a number of significant advantages:

- Flexibility: P4 makes many packet-forwarding policies expressible as programs, in contrast to traditional switches, which expose fixed-function forwarding engines to their users.
- Expressiveness: P4 can express sophisticated, hardware-independent packet processing algorithms using solely general-purpose operations and table lookups. Such programs are portable across hardware targets that implement the same architectures (assuming sufficient resources are available).
- Resource mapping and management: P4 programs describe storage resources abstractly (e.g.,IPv4 source address); compilers map such user-defined fields to available hardware resources and manage low-level details such as allocation and scheduling.
- Software engineering: P4 programs provide important benefits such as type checking, information hiding, and software reuse.
- Component libraries: Component libraries supplied by manufacturers can be used to wrap hardware-specific functions into portable high-level P4 constructs.
- Decoupling hardware and software evolution: Target manufacturers may use abstract architectures to further decouple the evolution of low-level architectural details from high-level processing.
- Debugging: Manufacturers can provide software models of an architecture to aid in the development and debugging of P4 programs.

A.3 Example: A very simple switch

As an example to illustrate the features of the P4 architecture, consider implementing a very simple switch in P4. This example demonstrates many important features of the P4 programming language. This architecture named the “Very Simple Switch” (VSS). Fig.6.6 is a diagram of this architecture. VSS has a number of blocks; those in cyan are fixed-function blocks , and the white blocks are programmable using P4.

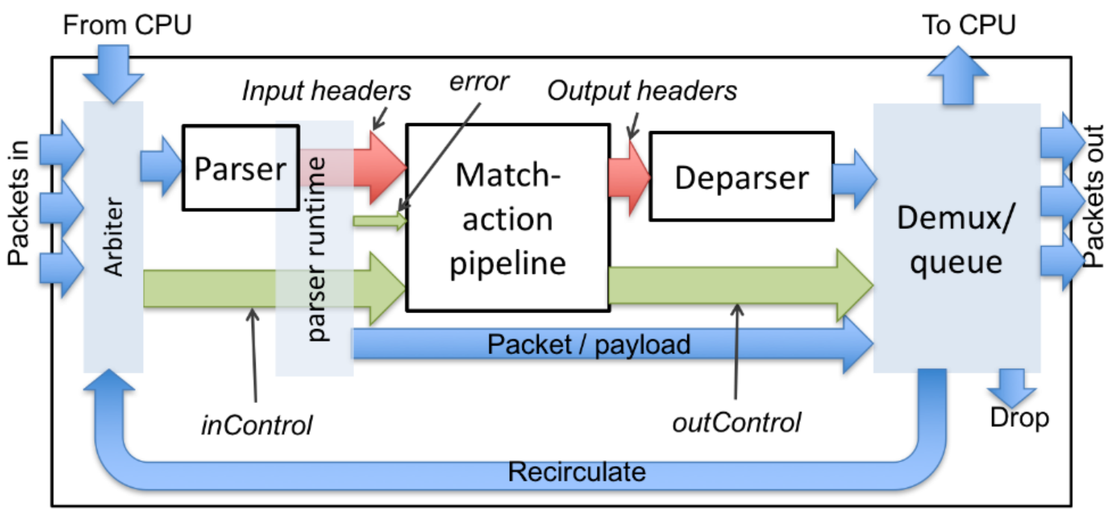


Figure 6.6 – The Very Simple Switch (VSS) architecture [4].

VSS receives packets through one of 8 input Ethernet ports, through a recirculation channel, or from a port connected directly to the CPU. VSS has one single parser, feeding into a single match-action pipeline, which feeds into a single deparser. After exiting the deparser, packets are emitted through one of 8 output Ethernet ports or one of 3 "special" ports:

- Packets sent to the "CPU port" are sent to the control plane
- Packets sent to the "Drop port" are discarded
- Packets sent to the "Recirculate port" are re-injected in the switch through a special input port

The white blocks in the figure are programmable, and the user must provide a corresponding P4 pro-gram to specify the behavior of each such block. The red arrows indicate the flow of user-defined data. The cyan blocks are fixed-function components. The green arrows are data plane interfaces used to convey information between the fixed-function blocks and the programmable blocks—exposed in the P4 program as intrinsic metadata.

Very Simple Switch Architecture

The following P4 program provides a declaration of VSS in P4, as it would be provided by the VSS manufacturer. The declaration contains several type declarations, constants, and finally declarations for the three programmable blocks. The programmable blocks are described by their types; the implementation of these blocks has to be provided by the switch programmer. To have more details about the VSS architecture, refer to [4].

```
// File "very_simple_switch_model.p4"
```

```
    // Very Simple Switch P4 declaration
    // core library needed for packet_in and packet_out definitions
    # include <core.p4>
    /*Various constants and structure declarations*/
    /*ports are represented using 4-bit values*/
    typedef bit<4> PortId;
    /*only 8 ports are "real"*/
    const PortId REAL_PORT_COUNT =4w8;// 4w8 is the number 8 in 4 bits
    /*metadata accompanying an input packet*/
    struct InControl {PortId inputPort;}
    /*special input port values*/
    const PortId RECIRCULATE_IN_PORT =0xD;
    const PortId CPU_IN_PORT =0xE;
    /*metadata that must be computed for outgoing packets*/
    struct OutControl {PortId outputPort;}
    /*special output port values for outgoing packet*/
    const PortId DROP_PORT =0xF;
    const PortId CPU_OUT_PORT =0xE;
    const PortId RECIRCULATE_OUT_PORT =0xD;
    /*Prototypes for all programmable blocks*/
    /***Programmable parser.
    *@param <H> type of headers; defined by user
    *@param b input packet
    *@param parsedHeaders headers constructed by parser*/
    parser Parser<H>(packet_in b,out H parsedHeaders);
    /***Match-action pipeline
    *@param <H> type of input and output headers
    *@param headers headers received from the parser and sent to the deparser
    *@param parseError error that may have surfaced during parsing
    *@param inCtrl information from architecture, accompanying input packet
    *@param outCtrl information for architecture, accompanying output packet*/
    control Pipe<H>(inout H headers,
                    in error parseError,// parser error
                     in InControl inCtrl,// input port
                     out OutControl outCtrl);// output port
/***VSS deparser.
*@param <H> type of headers; defined by user
*@param b output packet
*@param outputHeaders headers for output packet*/

    control Deparser <H> (inout H output Headers, packet_out b);

/***Top-level package declaration - must be instantiated by user.
*The arguments to the package indicate blocks that
*must be instantiated by the user.
*@param <H> user-defined type of the headers processed.*/
```

```
package VSS <H> (Parser<H> p, Pipe<H> map, Deparser<H> d);

// Architecture-specific objects that can be instantiated
// Checksum unit
extern Checksum16 {
    Checksum16();// constructor
    void clear();// prepare unit for computation
    void update<T>(in T data);// add data to checksum
    void remove<T>(in T data);// remove data from existing checksum
    bit<16> get();// get the checksum for the data added since last clear}
```