Y.K. TUNCEL

ÇANKAYA UNIVERSITY

APPLICATION LAYER PROCESSING WITH PROTOCOL INDEPENDENT SWITCH ARCHITECTURE

YUSUF KÜRŞAT TUNCEL

FEBRUARY, 2021

APPLICATION LAYER PROCESSING WITH PROTOCOL INDEPENDENT SWITCH ARCHITECTURE

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
ÇANKAYA UNIVERSITY

BY
YUSUF KÜRŞAT TUNCEL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

FEBRUARY 2021

ABSTRACT

APPLICATION LAYER PROCESSING WITH PROTOCOL INDEPENDENT SWITCH ARCHITECTURE

Tuncel, Yusuf Kürşat
M.Sc., Department of Computer Engineering
Supervisor : Assist. Prof. Dr. Roya CHOUPANI
Co-Supervisor : Assoc. Prof. Dr. Kasim ÖZTOPRAK

February 2021, 98 pages

This thesis investigates and proposes a solution for Protocol Independent Switch Architecture in order to process application layer data, enabling the inspection and processing of application content. Protocol Independent Switch Architecture (PISA) is a novel approach in networking where the switch does not run any embedded binary code for processing of network packets but rather an interpreted code written in a domain-specific language. The main motivation behind this approach is that telecommunication operators do not want to be locked in by a vendor for any type of networking equipment, develop their own networking code in a hardware environment that is not governed by a single equipment manufacturer. This approach also eases the modeling of equipment in a simulation environment as all of the components of a hardware switch run the same compatible code in a software modeled switch. The novel techniques in this thesis exploit the main functions of a programmable switch and combine the streaming data processor software to process application layer data to create the desired effect from a telecommunication operator perspective to lower down the costs, achieve desired performance and govern the network in a comprehensive manner. The results indicate that the proposed solution using PISA switches with a stream processor enables application visibility and control in an outstanding performance. The experimental study indicates that without any optimization, the proposed solution increases the performance of application identification and control systems from 5,5 up to 47 times.

Keywords: Software-Defined Networks, Protocol Independent Switch Architecture, Programmable Switches, P4, Virtualization, Cloud-Native, Stream Processor, Deep Packet Inspection

ÖZ

PROTOKOLDEN BAĞIMSIZ AĞ ANAHTAR MİMARİSİ İLE UYGULAMA KATMANI İŞLEME

Tuncel, Yusuf Kürşat
Yüksek Lisans, Bilgisayar Mühendisliği Anabilim Dalı
Tez Yöneticisi : Dr. Öğretim Üyesi Roya CHOUPANI
Ortak Tez Yöneticisi : Doç. Dr. Kasım ÖZTOPRAK

Şubat 2021, 98 sayfa

Bu tez, uygulama katmanı verilerini işlemek ve bir uygulama içeriğinin incelenmesini sağlamak için Protokolden Bağımsız Anahtar Mimarisi için bir çözüm araştırır ve bir yöntem önerir. Protokolden Bağımsız Anahtar Mimarisi, ağ anahtarının herhangi bir gömülü ikili kod çalıştırmadığı, bunun yerine amaca özel bir dilde yazılmış yorumlanmış bir kod çalıştırdığı ağ iletişiminde yeni bir yaklaşımdır. Bu yaklaşımın arkasındaki ana motivasyon, telekomünikasyon operatörlerinin herhangi bir ağ ekipmanı türü için bir satıcıya kilitlenmek istememeleri, tek bir ekipman üreticisi tarafından yönetilmeyen bir donanım ekosisteminde kendi ağ kodlarını geliştirmeleridir. Bu yaklaşım aynı zamanda, bir donanım anahtarının tüm bileşenleri yazılımla modellenen bir anahtarda aynı uyumlu kodu çalıştırdığı için bir ekipmanın simülasyon ortamında modellemesini kolaylaştırır. Bu tezdeki yeni teknikler, programlanabilir bir anahtarın ana işlevlerinden yararlanarak, maliyetleri düşürmek ve ağı kapsamlı bir şekilde yönetmek için bir telekomünikasyon operatörü perspektifinden istenen etkiyi yaratarak akışlı veri işlemci yazılımını ağ anahtarı yazılımıyla birleştirmeyi amaçlamaktadır. Deneysel çalışma, herhangi bir optimizasyon yapılmadan önerilen çözümün uygulama tanımlama sistemlerinin performansını 5.5'ten 47 katına çıkardığını göstermektedir.

Anahtar Kelimeler: Yazılım Tanımlı Ağlar, Protokolden Bağımsız Anahtar Mimarisi, Programlanabilir Anahtarlar, P4, Sanallaştırma, Bulutta Yerel, Akış İşlemcisi, Derin Paket Denetimi

ACKNOWLEDGMENTS

I wish to express my deepest gratitude to my supervisors Assist. Prof. Dr. Roya Choupani and Assoc. Prof. Dr. Kasım Öztoprak for their guidance, advice, criticism, encouragement, and insight throughout the research. Your insightful motivational comments encouraged me throughout this process.

My sincere acknowledgment also goes to my thesis committee members Prof. Dr. Mehmet Reşit Tolun for his great contribution in proof reading of this thesis and Assist Prof. Dr. Abdul Kadir Görür for their motivation as well as guidance.

This thesis would not be possible without the technology developed by A. Elbakyan.

I would like to thank Dr. İsmail Bütün for his careful review and helpful comments.

I also would also like to thank my wife and daughter for their support, understanding, and motivation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AI/ML	Artificial Intelligence/Machine Learning
ASIC	Application-Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
DPI	Deep Packet Inspection
eNB	E-UTRAN NodeB
ETSI	European Telecommunications Standards Institute
HW	Hardware
IP	Internet Protocol
MEC	Mobile Edge Computing
MPLS	Multi-Protocol Label Switching
NFV	Network Functions Virtualization
ONAP	Open Network Automation Platform
OSM	Open Service Management
PISA	Protocol Independent Switch Architecture
RAN	Radio Access Network
SDN	Software-Defined Networks
SW	Software
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
ZSM	Zero Touch Networking and Service Management

CHAPTER 1

INTRODUCTION

1.1. Introduction

Rapid technology changes also affected operators in the telecommunication world. As an artifact of this dramatic change, the operators face several issues which are not limited to the following: i) Traffic is growing quick, ii) Capex and Opex tracking traffic growth and not declining fast enough, and iii) Revenue is flat or declining. All these changes threaten the viability of their business. Besides, they would not meet the enormous increase in traffic demand with traditional networking infrastructures and services.

During the last two decades, desktop and server virtualization has played an active role in the Information Technology (IT) world. We are still in the middle of the transformation, while all parties have experienced the effect of better resource utilization and ease of usage. Similarly, when Openflow [1] was the first building block of the Software-Defined Networking (SDN) enabling communication with switches, most of the world was unaware of the birth of a revolution in the telecommunication world. The proposed solution was revisiting the early telephony networks with a clear separation of control and data planes. This is not surprising since almost all significant revolutions (e.g., ATM, MPLS, and so on) in the telecommunication world were revisiting original telephony networks' ideas to simplify network management with better service quality to optimize the costs in the operations.

Contemporary to that progress, multiple government-funded projects started to simulate large networks through server infrastructure. Because of these efforts, controlling network devices is extended to controlling physical and hypervisor-based virtual network devices. Successfully separating the control plane and data plane by defining a communication protocol, the need for the intelligence aroused to perform the network wise decisions and enforce the determined communication protocol to harmonize the networking. This demand is met by introducing SDN controllers.

The enormous progress in clarifying the picture in the control plane sped up the data plane's workings. The glue fulfilling the evolution in networking and SDN was the network virtualization to adapt itself to existing heterogeneous hardware. Contemporarily virtualization is considered, and Network Function Virtualization (NFV) is introduced to utilize the specialized networking boxes. ETSI thinks that SDN and NFV are complementary to each other [2]. NFV will allow the operators to replace the appliances for network functions such as firewalls, load balancers, and customer premises gateways by virtually delivering those services [3]. [4] is adding openness (mainly open source) as the third pillar to SDN and NFV.

With the use of OpenFlow, SDN, and NFV, the network revolution has started. However, it did not mean much to the telecommunication operators because of the demands of their business. Something was missing in the picture to motivate them to use, or at least the gaining from the new solutions did not get massive interest from them with their implementation of the services and architecture of their world. The operator world was designed to have vertical SILOs with their management domains, which is not practical since they would have tens of thousands of more devices than they had in the past.

The operators' missing part in the transformation was the Policy Management and Orchestration systems with the capabilities of delivering new services and allocating resources dynamically upon the demand of the users and systems, and applying the policies defined by the operator. The need for an umbrella to enable harmonization between SILOs was evident by the operators. The operators realized the efforts of having such solutions. They started a series of projects to fill the missing part of the transformation in order to brighten the telecommunication world's future.

As stated previously, telecommunication world is in a great transformation. The most important aspect of this transformation is to switch from old hardware-dependent, vertical architectures to software-defined architecture. In this architecture, there are series of improvements compared with the current products. Although the use of NFV was a key improvement in data plane with improved flexibility, Protocol Independent Switch Architecture (PISA) is one of the key elements with the accelerated performance and intelligent processing ability in the data plane during this change. The change in the architecture affects all stakeholders in a telecommunication operator infrastructure including applications. Legacy Applications written for legacy hardware are transformed into Software-defined architecture.

Independent from the Software Defined Architectures, application identification and control became critical in the last decade. It perched itself into the center of cybersecurity, accounting, quality of service management and similar services. One of the most important problems incurred by application identification is resource-hungry behavior of itself. Next-Generation Firewalls (NGFW) and Deep Packet Inspection (DPI) systems are two of the most popular usage area of application identification and control. DPI, as the name implies, inspects every packet that is running through the network deeply, and try to classify it under a human-readable name. It not only relies on packets metadata and header but also packet payload, hence the name “Deep”.

While L4 (OSI Layer-4) provides valuable information about a packet, it cannot give us any clue about the payload. In order to that, packets must be inspected by maintaining the stateful information, and the payload must be constructed accordingly so that it can be classified correctly. With the help of L4 information, network-side security, such as stateful firewalls can be built. Similar to NGFWs processing packets in L7, DPI still needs to inspect at L7. With the emergence of SDN architecture, DPI vendors switched from hardware to software-based L7 DPIs. As they switch from hardware-dependent architecture to SDN-based architecture, they lack the proper scalability to match the actual line speed of the switches. While the capacities of the data backbone increase, the systems depending on application identification became the bottleneck of the infrastructure.

As explained before, the network applications become Virtual Network Functions (VNF). Current software-based DPI systems (DPI VNFs) can scale up to 100 Gbps in a Virtual Machine running on top of powerful hardware. As the demand increases, the telecom operators will need application identification systems like NGFWs and DPI systems running with the speeds in the order of Tbps of traffic classification in real-time and such as in a single instance of DPI. The performance gain arises from the fact that the classification operation starts at the switch-level code data plane and continues in the user-plane.

1.2. Thesis Contribution

In this thesis, we aimed to introduce the application layer processing capabilities of P4-based programmable switches and their usage in application layer processing. We investigated and proposed a solution for Protocol Independent Switch Architecture in order to process application layer data, enabling the inspection of application content and triggering an appropriate response. Protocol Independent Switch Architecture is a novel approach in networking where the switch does not run any embedded binary code but rather an interpreted code written in a purpose-specific language. The main motivation behind this approach is that telecommunication operators do not want to be locked in by a vendor for any type of networking equipment, develop their own networking code in a hardware environment that is not governed by a single equipment manufacturer and single code base. This approach also eases the modeling of equipment in a simulation environment as all of the components of a hardware switch run the same compatible code in a software model to help researchers develop their code and simulate it without access to actual P4 hardware. The novel techniques in this thesis exploit the main functions of a programmable switch and combine the streaming data processor for application layer data processing software to create the desired effect from a telecommunication operator perspective to lower down the costs and govern the network in a comprehensive manner.

In this study, it is aimed to display the change from SILO oriented legacy telecommunication services to the future of telecommunication systems, including subscriber-defined services with near real-time processing and very low delay and higher capacities without limitation to the access platform. Many researchers performed studies in the area of progress and effect of SDN and NFV partially [3], [4], [5–14] however, none of the studies present an end-to-end picture defining all aspects for an operator as well as defining the application layer processing of programmable switches. The study differs from the others by two critical elements: i) it covers the whole picture for an operator from an inside view of an operator, and ii) it presents the progress of all fields in the picture covering progress in the control plane (SDN), data plane (NFV, PISA and others), and evolution of software development culture with the improvements in orchestration and automation. The effect of programmable hardware and almost human-free operation is introduced and elaborated. Although the research and adaptation are spreading among the operators, most decision-makers and adapters are confused about what SDN is and what it will bring. There are also some PR efforts by some vendors, trying to position SDN as a magic pill to solve all the existing problems of the operators. Indeed, it would even increase the complexity of the issues without proper planning before production.

This study proposes a solution using PISA switches with an application layer stream processor enabling application visibility and control in an outstanding performance.

The proposed architecture processes the packets in a network switch while selecting only necessary ones to the L7 based systems such as DPI and NGFW. This approach increments the performance of NGFW and DPI systems in the order of 40 times. Building such flexible and scalable application visibility system is challenging. Achieving this goal brings a question: How network operators should design such solution processing packets in L7 knowledge with the performance of L4, in other words, they should figure out how to scale out such system for high volume of data in real-time? One will find out the answer to this question throughout this thesis.

1.3. Thesis Organization

Chapter 2 gives a background on the change in a Telecommunication operator world with the use of Software-Defined Networks and their application in Telecommunications networks in general. The literature summary points out that the change is not limited to a single improvement in SDN, rather than it depicts that it is a change in the culture of software development, architectural design, and approach to the subscribers affecting all the stakeholders of the telecommunication systems.

Chapter 3 explains the proposed system architecture, bringing data plane performance into L7 systems (such as DPI and NGFW) by using our approach for the thesis.

Chapter 4 discusses the experimental study, gives the results of the experimental study.

Chapter 5 concludes the thesis by elaborating the results of this study together with the transformation of telecommunication systems. This part also points to the importance of the contribution of this study into the literature and telecommunication operator and their vendor ecosystem.

CHAPTER 2

BACKGROUND

2.1. Introduction

Software-Defined Networks (SDN) and its practical applications draw great interest from researchers in the telecommunication field. SDN separates control and forwarding planes of a network and provides a centralized, simplified view for improved automation and orchestration of services. SDN controllers provide communication between network planes that have been separated by a networking device. NFV concentrates on the software dedicated to networking functions. NFV separates network services - including firewalls, content storage, name lookup, routing, and load balancing - from vendor equipment. Separated services can be executed in a virtualized environment to innovate and quickly provision services due to the cloud. This separation allows the flexibility of selecting/defining services for a subscriber forming chain of services which is called service function chaining (SFC). NVF ensures the network can seamlessly integrate with multiple virtualization technologies, particularly those that support multi-tenancy.

Switches and routers are presented as white boxes that are made from off-the-shelf standard chipsets in an open market, compared to proprietary chipsets that are designed by an individual producer. Hence, networking software and protocols can be deployed and executed via SDN without the constraints of working with one particular equipment manufacturer’s proprietary restrictions.

The research topic has numerous fields, as depicted in Table 1, that capture the main search areas in SDN. The topics in SDN draw enormous attention from telecommunication sector as a top area of interest. The reasons for this interest are going to be explained in the next section.

Table 1 Keywords used to search in Google Scholar and Microsoft Academic, between 2016

Keywords	Google	Microsoft Academic	Type
SND NFV (without quotes)	21100	3555	Main
"SDN NFV" (with quotes)	8250	372	Main-Backup
"Network Automation"	4900	283	Mixable
"Edge Computing"	42200	10655	Mixable
"Openflow"	25200	3524	Mixable
"Cloud-Native"	5270	223	Mixable
"DevOps"	17400	1541	Mixable
"Zero Touch Network"	181	14	Single
"P4 language".	859	51	Single
"Edge Computing" SDN NFV	5260	214	Combined
"Openflow" SDN NFV	8210	1520	Combined
"Network Automation" SDN NFV	910	275	Combined
"Cloud-Native" SDN NFV	676	214	Combined
"DevOps" SDN NFV	689	1289	Combined

Type field contains 5 types:

Main and Main-Backup: Primary search terms, combined or separately.

Mixable: Search term that can be used together with the main term.

Single: Search term that cannot the used together with the main term.

Combined: Search term combined with the main term.

2.2. The Need for Change in the Telecommunication Sector

Before discussing further details of SDN, NFV, and their impact, it would be better to define the current status and future expectations from an operator roughly. As depicted in Figure 1, in a traditional operation, the operator's principal assets are the transmission infrastructure bringing the connectivity between the core cloud and the broadband access for the subscribers. Typically, the operators are hosting their compute infrastructure, mainly for hosting OSS, BSS systems, and some services through data centers in the core cloud. The subscribers are accessing communication services through broadband access, while an IP communication network forms the backbone. The design is simple, and the only purpose of broadband access is to connect the subscribers to the services. The compute technologies provided by a conventional operator are aggregated into a few data centers. It is also possible to quantify the service delivered currently by such operators: i) broadband access bandwidth in 10s of Mbps with a typical latency around 100ms. The number of connected devices is around 10 Billion. The numbers will change dramatically in the order of 100x in the new era of telecommunication. The communication speed becomes in the order of Gbps while the latency is targeted to be 1 ms and 100s of Billions of connected devices.

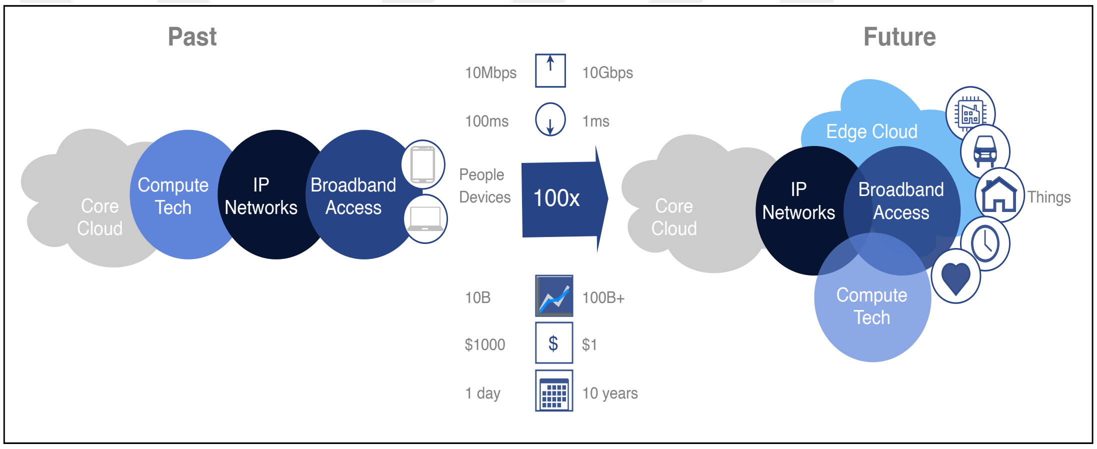

Figure 1: The Change in The Capacity Demand, Latency, and Services in the Telecommunication World

Any comparison concentrated on the numbers' change will be an injustice to the new era of the telecommunication world, especially for the services. The researchers [3] summarize the value proposition of SDN and NFV technologies to the operator business as having; i) virtualized, programmable, and scalable networks, ii) automated provisioning and configuration, as well as centralized control and management, and iii) differentiated and agile services with simpler provisioning and higher revenue generation efficiently integrating into third-party systems.

The architecture of the operators will change slightly in the new era. The computing technology will shift from the core data centers towards the Edge as well as forming a new edge cloud through a new generation mobile access cloud beside the computing power. While the core cloud, IP backbone, and access will exist in their position, their structure will transform. First, the data centers will have extensions with enough computing power at the Edge delivering near real-time processing power, especially for systems with the need for low latency like autonomous devices, IoT, and caching the traffic intelligently to reduce the traffic load through the network. This also triggers the profile of the users accessing the services heavily from the people to the things. The new architecture conforms to the telecommunication world's catchword: ***distribute when you can, centralize when you must***.

Leading operators in the telecommunication world started to adopt SDN into their network to build a network infrastructure that will optimize costs and spin out new services faster than the current situation for their customers. NFV is the complementary technology in creating the target telecommunication architectures for SDN-NFV transformation.

The change in the telecommunication systems is not limited to the infrastructure's architecture but mainly focuses on the way of approaching the subscribers. The systems are changing to be customer-oriented service-based systems rather than system-oriented subscription basis. This change in subscribers' approach ultimately needs end-to-end automation from service requests to delivery, including provisioning, maintenance, and service closure. The adaptation of SDN and NFV is transforming the networking and changing the culture and roles of people in the IT chain, and similar to what DevOps brought to software development and operation life cycle. Telecommunication networks are transforming to become a New Infrastructure paradigm than merely extending the Cloud by considering the evolving "Edge" demands. This shift is not limited to but includes the adaptation of edge computing, which becomes mandatory for 5G and IoT applications in real-time, as well as extending their edge to customer premises through SD-WAN.

All these improvements are evolving together with advances in software development culture. The evolution of networking technologies is triggered by the progress in software development culture and information technologies. The building blocks enabling the transformation in the IT world are explored in the next section.

2.3. Evolution of Information Technologies and Related Environments

While the networks transform into software-driven, programmable, service-based infrastructures, the new paradigm should provide agility, scalability, and fully automated systems. Such phenomena in the world are always triggered by cultural changes in the way of doing the tasks. One of the most significant moves in this paradigm is the cultural change in application development, deployment, and maintenance and operating it in a new mindset.

The movement from monolithic application development to microservices changed the structure of all stories. It started with the commencement of the cultural movement of application development named by Debois, P. as DevOps [5]. Although there are some formalized set of operational processes defining the workflow and relationship of service design, strategy, transition, operation, and continual service improvement like IT Infrastructure Library (ITIL), DevOps focuses on the productive collaboration of software developers and IT operation personnel by changing attitudes, processes and team interactions [6]. Unlike ITIL, there is no clear delineation in DevOps. It uses agile software development methodologies and applies them to automate all software lifecycle steps from development to deployment for operation. DevOps broke the burdens of traditional SILOs and brought agility into the whole application and operation lifecycle.

2.3.1. VMs, Containers, Dockers, and Kubernetes

Another significant movement affecting this phenomenon happens around virtualization technologies. The networking side's transformation has a tightly coupled relationship with the one on the server-side, especially in NFV. The story started at the Massachusetts Institute of Technology (MIT) for the MAC project, which stood for Mathematics and Computation [7]. The project's needs gave rise to developing the first time-sharing operating system (OS) to utilize all the computer resources.

In the late 80s and early 90s, virtualization took the role of running a different operating system on top of a host OS to help the users for software compatibility. Although virtualization was to get the ability to run an application on other hardware platforms such as running windows (and its applications) on a mainframe environment, later, it turned out to be a resource utilization-oriented approach to lower the costs. In the late 90s, VMWare [8] became the flagship of computing resources' virtualization, including servers and desktops. Virtual desktop environments were another flavor of this progress in allowing the users to use a desktop running on the server-side, which simulates early time-sharing environments with a sophisticated graphical user interface. A generic VM architecture has a hypervisor controlling all the infrastructure resources to serve guest operating systems, as shown in Figure 2. Every virtual machine allocates separate memory and computes resources for its OS; thus, it should keep the copy of a packet for itself while processing, which is the reason for performance degradation. These performance problems and additional security concerns resulted in the development of containers.

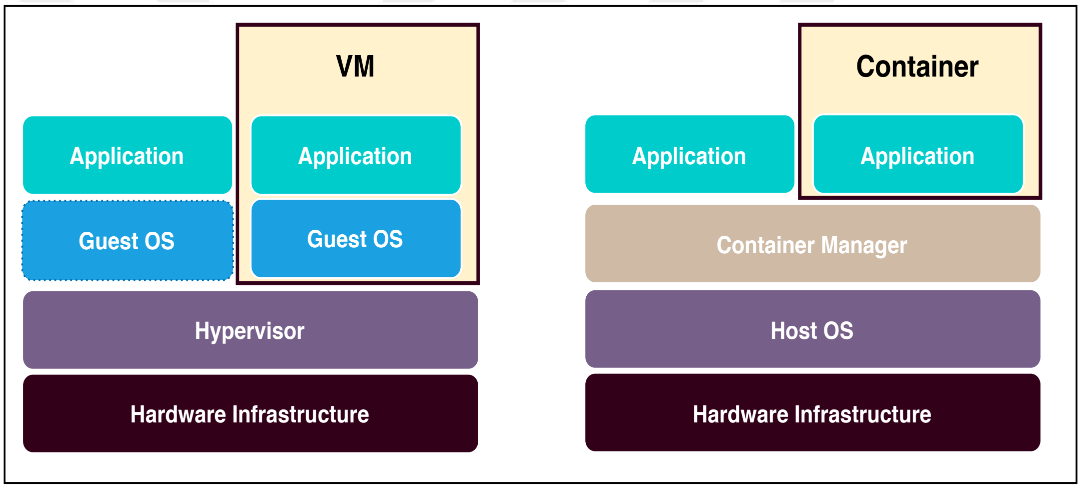

Figure 2: Virtual Machines Compared with Containers

Unlike VMs, containers are built on a single operating system and managed by a container manager, as illustrated in Fig. 2. Shared components, among other containers, are read-only libraries. Container-based applications can be started in the order of seconds compared to a few minutes in VM-based applications. Containers increase the use of shared resources much higher than VMs, reducing private parts of resources. This nature of the containers makes them lightweight, resulting in higher granularity in a machine and higher performance than VMs.

There are some tools facilitating containers by allowing users to create, deploy, and run applications like Docker [9]. In addition to more accessible packaging support, it also has a clustering tool called Docker Swarm to schedule and orchestrate clusters of containers in different machines. Similarly, Kubernetes is developed at Google to automate the deployment, scheduling, and scaling of containerized applications and support many containerization tools such as Docker [10]. Later it has been donated to Cloud-Native Compute Foundation (CNCF) under the Linux Foundation as an open-source project. It is the de-facto standard in container orchestration with the capabilities of grouping containers into logical units allowing the systems to distribute containers into multiple physical nodes scaling up to enormous dimensions and load balancing. By using Kubernetes, the big data center owners or operators can run multiple instances of an application and independent upgrades and versioning. On the NFV side, container technology's effect transforms the VNFs into Container Network Functions (CNF).

In comparison, while containers enable isolation of performance by managing the CPUs, memory, and similar resources, Docker allows easier control and packaging. On the other hand, Kubernetes stays at a higher level dealing with the composition of services, load balancing, naming the services, and controlling multiple versions of services (by enabling the versioning). Kubernetes handles all those and routes the tasks to the proper implementations. This ability at a higher level of orchestration allows the users to decouple operations from deployment to have a granular separation of the functions.

The researchers conducted several studies to demonstrate the performance comparison of virtual machines and container systems. [11] conducted a performance testing between Linux's kernel-based virtual machine (KVM), Docker, Linux Containers (LXC), and Cloud Operating System (OSv). The tests were performed on comparing CPU, Memory, Disk I/O, and Network I/O performance degradation from native usage. The container-based solutions (Docker and LXC) outperform VM-based solutions similar to the study of IBM research, in which they report similar results in[12]. Lately, the analysis compares in application domains, such as big data [7] using Spark, NoSQL environments [13] using Cassandra. Both studies show that container-based solutions outperform virtual machine-based solutions with better resource utilization and scalability. In addition to performance, security is another concern in IT systems, as mentioned in [14] where container-based methods are evaluated more secure by minimizing attack vector compared to VM-based solutions.

Although the change in the virtualization methods improves the performance of the systems, as can be seen from serious studies[11][12][14], the change in the virtualization is a revolution in; i) concentrating on the services rather than machines and structuring them as microservices, ii) dynamically managing services by scaling, updating, and co-operating multiple versions as needed, and iii) grouping the containers to simplify management, enable load balancing, high availability, and deployment. This different approach comes from containers' power in opening resource-efficient services quickly and retiring them similarly when the demand expires and maintains resource and security isolation between services.

2.3.2. Cloud-Native, Edge Computing, and Microservices

The cultural change with DevOps processes brought agility, effective use of microservices architectures, and continuous integration and development (CI/CD) workflows. Containers became the best suitable implementation platform for those microservices-based, agile applications. This total change enhanced the applications to run in a cloud environment smoothly. Lately, the cloud-native term is widely used to define the applications and services capable of running in cloud environments. While cloud-native services focus on overall user experience and the companies' internal IT compliance, cloud-native services are focusing on delivering to the massive scale of applications. This nature simplifies cloud-native applications because of their small and stateless nature, which increases mobility and scalability. Cloud-native services span across the data centers, Edge, and user devices. A Cloud-native mindset is a key to leveraging compute at the Edge [15]. The architectures leverage the accountability for computing and communication while bringing intelligence towards the Edge.

The customers' changing demands helped shift the focus of service providers from traditional virtualization-based data centers to simple, flexible, microservice-based cloud-native services using Linux containers [15]. This change allowed service providers to speed up the rapid development and deployment of new services to scale up and down upon changing demands and traffic patterns.

2.3.3. Data Centers

The data centers evolved in several generations. In the first generation, they aimed to optimize the cost of capital expenditures, mainly in hardware, by using virtualization.

The second generation of data centers, currently in use, uses public clouds, which frees enterprises to set up their physical systems. The business logic is still the same and manages all the systems remotely. The evolution from the first generation to the second generation can be summarized as changing from hardware to software-based data centers.

The future generations will be server-less computing. In this generation, one does not deal with today's daily tasks such as OS configuration, load balancing, or patching systems, but only needs to deal with the computing capacity or services. The customers' objectives will be the quality of services throughput or latency of the applications or functions rather than several VMs or the amount of storage.

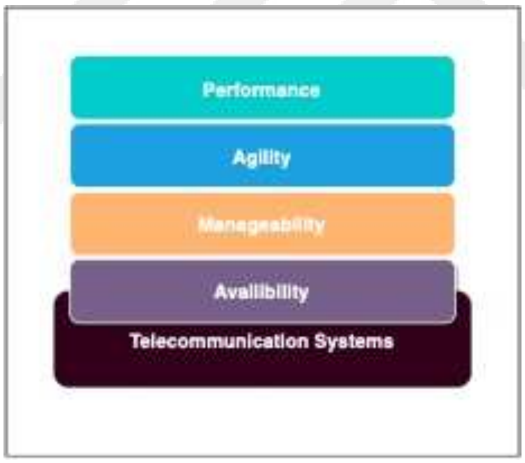

Figure 3: Four features of a Next Generation Telecommunication System

2.4. Expectations of the Operators

The new demands from telecommunication systems and technological progress lead the telecommunication research community to prioritize new features for the target system designs. Figure 3 summarizes the four critical areas that the research is concentrating on.

In the past, most of the operators were only concentrating on the availability of the services. The box-oriented approach in telecommunication infrastructure resulted from this basic approach favoring simplicity. It was easier to implement a single application without considering integrating it into other systems. Availability is still the first expectation from a system to guarantee stability and design with security concerns. An insecure platform is unavailable since someone else shares the platform's control with immediate access to the systems. The redesigned systems are trying to disaggregate the systems' parts to reduce the complexity of the systems that will also increase the availability. Disaggregation will be the key to improving availability.

Although availability is crucial for the services' existence, there was limited manageability support of the box-oriented vertical systems. Manageability brings visibility to systems, without which the performance does not matter. Automation is the part or mutated form of management favoring agility while designing new services and quickly delivering service to a customer. Automation is crucial while it would reduce the flexibility in the availability by enforcing the systems to comply with some standards or way of doing their tasks, which would lead them to lose some functionalities or performance.

The new drivers of technology increased the operators, and all kinds of providers' demands, squeezing them between the walls of needs and delivery. The traditional vendors were delivering new features or bug fixes in several months' even years according to their release cycles while the demands are evolving too fast. There is no future with the conventional way of doing business, and all providers should adopt new services or upgrades within at most a couple of weeks to survive.

With the burden of this evolution, the future of the integrated telecommunication world covering cloud, edge computing, and data centers, including cloud-native applications, services, and portability of them, will be built by using the open-source tools tightly integrating and orchestrating across containers. This evolution in open-source technologies shifting all communication systems into the next phase also requires some arrangements on compliance (regulations) and security protocols with being fully auditable to ensure providing acceptable service level and supporting common identity and access management, policy management, and a full range of service portability [16][17][18][19].

2.5. The Results of the Changes in the Operators

The operators are developing their products to survive within the competitive environment in a "saturated market". This allows them to deliver -any feature, any time- reducing the service time of a new service or a new feature in a current product to almost zero while having it with incredibly cheaper costs. The operators are becoming part of open-source initiatives to have flexible and less expensive systems and assuring security across the network.

Most of the open-source projects focus on either fiber or wireless access. In both product lines, while the intelligence was embedded in access equipment in the legacy products, it is taken out to the outside of the systems by disaggregation in the new telecommunication world, resulting in simple hardware with more sophisticated software. This idea is supported further by transforming many monolithic applications such as eNB into disaggregated control and data units with network slicing support. One key in this journey is disaggregation, which requires a redesign of network HW and SW [20]. Disaggregation allows researchers to work with a smaller subset of a problem at a time, leading to a speed-up in innovation while helping the technology users optimize the resources they use (such as reducing the hardware usage). As a result, the network equipment evolved from a legacy black box to disaggregated programmable equipment with control-data plane separation. Cloud RAN is proposed to create a programmable world for 5G and its future successors. Its main aim is to reach optimized converged networks. The total target in the cloud RAN is to accommodate the expectation of future services. Several initiatives define the standards of cloud RAN, such as O-RAN, C-RAN, and X-RAN projects.

Disaggregation in the systems triggered the consolidation of anything. Operators having mobile and wireline services will consolidate the control planes in a simple system. Simplification of user plane will allow operators to manage users as a single entity regardless of the subscription diversity to the services. The convergence in the control plane will let the operators define end-to-end network slicing and even defining a computing resource use service as a combination of cloud and edge computing resources.

Several factors are forcing the operators to join this revolution. Although the cost of serving the customers seems to be the priority, it would not be fair to put it in the first place. A more important reason that should be noted is the time delay between developing a new feature in the systems and putting it in production. Once the operators use the software-driven networking, it will help them deliver a new feature to all users almost within a week. Hence, it is evident that open source brings far faster deployment, updates, and innovations.

To succeed in this movement, success should be demonstrated publicly. Hence, network operators publicly stating that they are transforming their networks into a platform for innovative services and build the "network as a platform" by using SDN/NFV/Cloud with disaggregation, open-source, white boxes to reduce Capex and Opex significantly [20]. More interestingly, those changes recall the saying by Sun Microsystems: "Network is the computer." This applies to our case now: "the operators' infrastructure is the computer."

2.6. The Impact of Automation

The final part completing the picture in a carrier-grade network is the coordinator working closely with the operation of the applications and services that generally run on the network and the underlying infrastructure. The industry uses the term orchestrator [15] to coordinate and manage all network and compute elements needed to deliver a virtualized network service, including provisioning.

In 2014, ETSI ISG NFV published the standards for NFV related operations, interfaces, and functional points to conform to different requirements [21]. Several open-source products in NFV management and orchestration (MANO) solutions like ONAP [22], OSM [23], Open Baton [24], Cloudify [25], OPNFV [26] are at the various stages of development, based on the released ETSI standards. The needs of an operator beyond the boundaries of NFV management. They need to automate service design and creation, service orchestration, inventory management, control loop automation, policy management, SDN controller orchestration, Hypervisor management, legacy system management and similar tasks.

The most promising network automation software among the orchestrators, Open Network Automation Platform (ONAP) by Linux Foundation Networking, is almost becoming the de facto standard for the real brain of the whole infrastructure for an operator. Although OSM itself is owned by ETSI, the maintainer of the standard, it lacks critical features such as Kubernetes support, PNF integration, edge automation, real-time analytics, network slicing, data modeling, homing, scaling, and network optimization, as shown by a recent study [27].

To leverage the automation, Cloud-Native Architecture is critical. Figure 4 shows a brief difference between VM Architecture and Cloud-Native Architecture, as demonstrated by Kapadia [28]. The demonstrations aim to show how easy to onboard 5G core and Next-Generation Firewall with Cloud-Native Network Functions (CNFs) using ONAP, OPNFV, and Kubernetes, along with the working demonstrations and end-to-end testing in a lab environment.

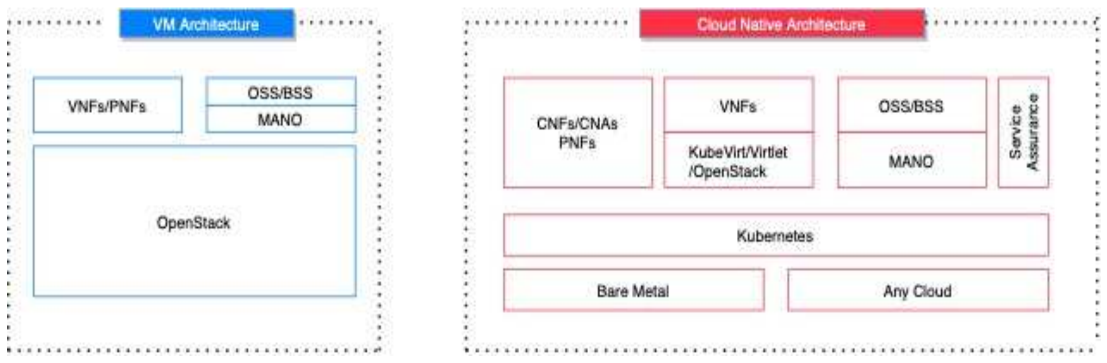

Figure 4: VM Architecture vs. Cloud-Native Functions

Going one step further, zero-touch networking and service management (ZSM) is proposed to get high-level human intents to generate low-level configuration generation for low-level devices and controllers with validated results. The main target of ZSM is to minimize the ratio of faults caused during human intervention.

2.7. Zero-Touch Networking and Service Management

There is a tradeoff in network operation between scalability, reliability, and efficiency. It is almost impossible to have all in the highest positive manner. This tradeoff results from an operator's basic requirements; i) enough capacity, ii) cheap infrastructure and operation, iii) high availability, and iv) rapid evolution to the changes.

According to Koley [29], 70% of all telecommunication systems' faults occur while being touched through management systems. Moreover, response times of provisioning procedures for techniques that will increase with 5G, such as edge computing and network slicing, will be required to be in the order of milliseconds, which indicate that manual operation with a human touch is out of the question.

The change in the architectures of delivery and operation triggered a new concept in the automation of all telecommunication services covering the entire lifecycle of network operations, including planning, delivering, onboarding, monitoring, updating, and decommissioning of services beyond the initial installation [15] which is called Zero Touch Networking and Service Management (ZSM) by ETSI [30]. ZSM has been receiving attention in the last years in this context, with no complete existing solution. Most studies focus on the models that can benefit from ZSM [31], [32], [33]. ZSM proposes a solution trying to keep all three elements together positively with intent-driven operation [34].

The main target of ZSM is to get high-level human intents to generate low-level configuration generation for low-level devices and controllers with validated results. Koley [29] proposes a Zero Touch Networking model designed to keep two infrastructure knowledge models: The network model and the configuration model. The network model and Configuration model are different views of the same information as topology and configuration. Once a change occurs in any of the models, it is reflected in the other model.

According to ETSI ZSM requirements based on documented scenarios [35], there are 39 scenarios in 176 total scenario requirements. One of the most challenging parts of these scenarios is called "Analytics & machine learning." Business requirements such as determining the root cause of a network anomaly and the ability to foresee network capacity exhaustion are few examples. To achieve these requirements, collecting a massive amount of historical and up-to-date network data and transferring it into a sandbox environment for self-learning are also defined in functional requirements. Once the analytics and learning capabilities are developed, they will be used in closed-loop automation.

The ambition towards AI/ML-based solutions for complex problems such as 5G management is not always easy to achieve. According to Benzaid and Taleb [31], although AI is seen as a critical factor for lowering operational costs and reducing the risk of human error, potential limitations and risks exist in using AI techniques. The authors summarize these limitations in 4 topics: i) Lack of Datasets and Labeling, ii) AI Model Interpretability, iii) Training Time and Inference Accuracy, and iv) Computation Complexity.

Similarly, the massive amount of telemetry data collected from network devices requires novel approaches and techniques to develop full-fledged, usable AI models. One of the most recent studies in this area [36], aims to solve the autonomous placement of Virtual Network Functions (VNFs) in 5G networks. Instead of using Supervised Learning (SL) models, the researchers used a particular form of Adaptive Reinforcement Learning. They achieved prediction accuracy performance gain by 40-45%, and overall VNF placement efficiency over against other SL benchmarks in 23 scenarios out of 27. This particular technique decouples the AI model from the training nodes, whereas other SL models are tightly bonded to the training nodes.

2.8. The Shift in Edge Computing

The term "edge computing" was first used by Akamai Technologies in 2002, which also holds a patent about it in 2004 [37]–[39]. In their context, edge computing was a particular methodology to deliver Java-based application content responsively to web-browsers to improve user experience.

The definition moved to Mobile Cloud Computing (MCC) after the popularity of Cloud Computing as a buzzword [40] [41]. MCC aimed to deliver content fast and efficiently to mobile users by using the infrastructure of the mobile operator or ad-hoc network created by the mobile users. Later, it is referred to as Mobile Edge Computing (MEC). A more recent and relevant definition from the chair of the MEC group of ETSI, Reznik, in the personal blog, was "anything that's not a "data center cloud" [42].

The most incentive that drives MEC is the enormous size of data and the computing power that the devices have to handle with the emergence of 5G networks. Mobile Edge Computing is capable of leveraging mobile resources by hosting computing applications, processing vast volumes of data prior to cloud sending, delivering RAN (Radio Access Network) cloud computing services in the last mile to mobile users. The applications that are empowered by MEC require immediate real-time responses, including but not limited to autonomous driving, telemedicine, remote surgery, robotics in production and warehouses, logistics, and many others. According to a recent survey on this area [43], "there are three main types of MEC use cases: consumer-oriented services, operator and third-party services, and network performance and QoE improvements."

The intelligence in Edge plays an important role in the delivery of services to consumers in close vicinity [44]. The researchers describe the transformation of MCC to MEC, stating the differences between the two and the driving forces of the transformation., which was renamed by ETSI as Multi-Access Edge Computing, dropping the “Mobile” part and extending the term to include fixed-mobile convergence [45]. The researchers provided a use-case for MEC to determine the proximity of a mobile user with the help of 5 different AI algorithms in comparison.

Telecommunication systems are not limited to wireless communications. Hence the edge in the wireline systems spans towards the customer premises. SD-WAN is the logical extension to the SDN infrastructure of the operators to customer premises. The revolutionary change at the core and access triggered SD-WAN's evolution to fulfill the customer's picture. The operators currently deliver simple L2 or L3 pipes as a VPN service with minimal traffic engineering support, mainly through their MPLS networks. The Edge in the future should support application-centric slicing and traffic engineering besides the current L2/L3 pipes. Another new improvement in the WAN side is Service Function Chaining (SFC).

SD-WAN [46] is one of the enablers of hybrid cloud ecosystems combining on-premise and cloud-based applications. SD-WAN solutions bring full flexibility to the customers' aggregating network functions from different vendors into a single box and enable the ease of access to the cloud. SD-WAN minimizes the need for MPLS between a central office and branch offices by using software-based techniques to reduce the need for high-speed connections, providing built-in packet-level security, deduplication, and data caching [47]. The study also demonstrates an SD-WAN network example using open-source software, which is OpenDayLight [48] [49]. According to Wu et al., there are at least 960 patent applications as of late 2020 containing SD-WAN as a keyword [50].

SFC is not limited to the edge of the telecommunication systems. Actively using SDN and NFV allowed operators to define a workflow for any kind of customer data flow through SFC's help at the core of their infrastructure. The ability to define customized paths for any data flow reduces the need for resources since only the prescribed flows pass through any network function contrary to the current deployments. All data flows pass through all functions residing in scalability and high resource consumption problems.

2.9. Next-Generation Security Services in Telecommunication Networks

Modern problems require modern solutions. As the SDN/NFV enabled networks and operators emerge, customers' cybersecurity services will be shaped differently shortly. In the foreword of "Guide to Security in SDN and NFV, the foreword author raises concerns about security in the SDN-NFV era by complaining the German presses suspicions that it "could be a tool for evil network operators to manipulate traffic flows against the public interest." [51] On the other hand, with the rise of IoT devices and a massive amount of data to deal with, SDN could be the best way to prevent IoT-based Distributed Denial of Service (DDoS) attacks [52].

Providing security-as-a-service could be one of the ultimate goals of telecommunication networks. As the processing power of network devices increases, the security services are moving towards the Edge. In a recent study with an attractive title, "Towards security-as-a-service in multi-access edge" [53], authors "propose a data-centric SECurity-as-a-Service (SECaaS) framework for elastic deployment and provisioning of security services at the Multi-Access Edge Computing (MEC) infrastructure." Motivated by the rapid growth of the Industrial Internet of Things (IoT), autonomous driving, and smart home applications, and the shortcomings of security measures taken at the core network to secure the services, authors suggest a novel security architecture that should be offered at the near edge of the network for tenants with different requirements by using the Named Data Networks (NDN) architecture [54] [55].

To offer security services, the underlying system architecture should be robust and secure as much as possible. In a recent survey on SDN-NFV security [56], authors conclude that at least three central issues and potential research areas are popular: i) The performance impact of enhancing security in SDN-NFV networks, ii) The importance of detecting abnormal behavior within the layers by monitoring, and iii) The security issues related with OpenStack.

2.10. Programmable Hardware

While SDN is the first half of the journey towards the programmable world, programmable hardware will build a dynamic system wholly programmable. In recent research on this topic, the authors of [35] explain the need for programmable hardware and the features of its' language in three items: "i) Reconfigurability in the field: Programmers should be able to change the way switches process packets once they are deployed, ii) Protocol independence: Switches should not be tied to any specific network protocols, and iii) Target independence: Programmers should be able to describe packet-processing functionality independently of the underlying hardware's specifics."

2.10.1. Protocol Independent Switch Architecture (PISA)

The research on programmable switches led to the definition of a reconfigurable match-action table (RMT) [57] based hardware that can be programmed with a domain-specific language. Protocol Independent Switch Architecture (PISA) is a special case of RMT, that supports P4 language as the default domain-specific language [58].

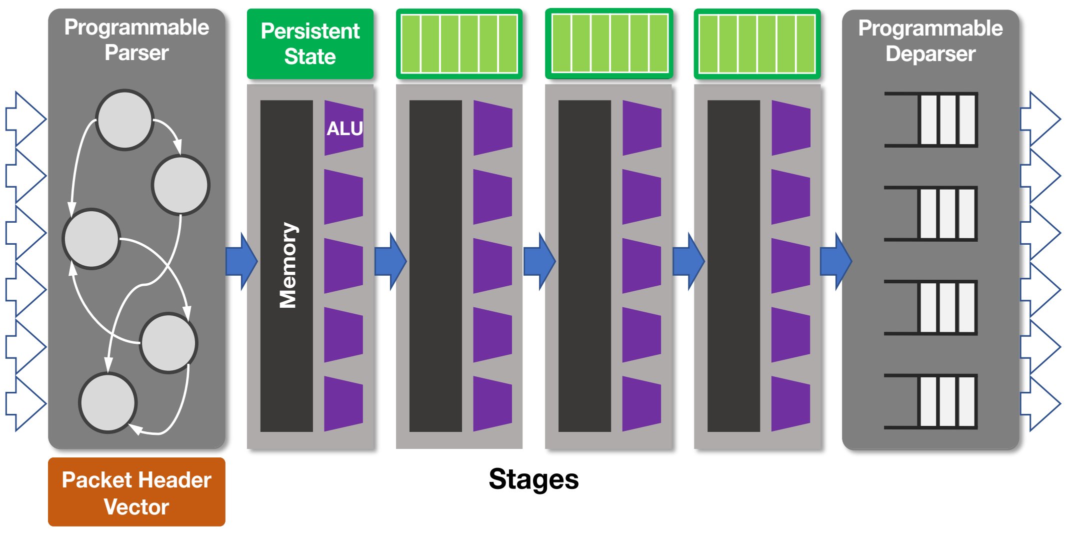

Figure 5: PISA Match-Action Table Processing Pipeline (Gupta et al., 2018)

A typical PISA switch consists of a programmable parser, ingress match-action table, a queue, a set of registers to keep the state of variable, egress match-action table and a programmable deparse as shown in Figure 5: PISA Match-Action Table Processing .

The parser and deparser are programmed for processing any type of header, specifically user-defined ones. The ingress and egress pipelines are the actual packet processing units that go through match-action tables in stages. Match-action tables match the header based on a set of rules that are controlled by the control plane and perform the corresponding action on the packet. Actions use primitives to modify the non-persistent resources (headers or metadata) of each packet.

2.10.2. P4 Language

Although there are several studies developing and using programmable hardware [59]–[62] ,the early use of programmable hardware is to make ease of use of telemetry data. Telemetry data is crucial for an automated future but generating telemetry data is not a trivial task. Adding more hardware and software to the routing and switching systems makes the current architecture more complex than ever. Since the telemetry data is generated at the packet level, the most logical way of doing this seems to be arising from the packet generating software at the hardware level, which leads us to P4, Programming protocol-independent Packet Processors, as called in the original paper defining it [63]. P4 is a domain-specific programming language for packet-processing hardware such as a router, switch, network interface cards, and network function-related appliances that work and data plane based on the decisions from the control plane as in Figure 6.

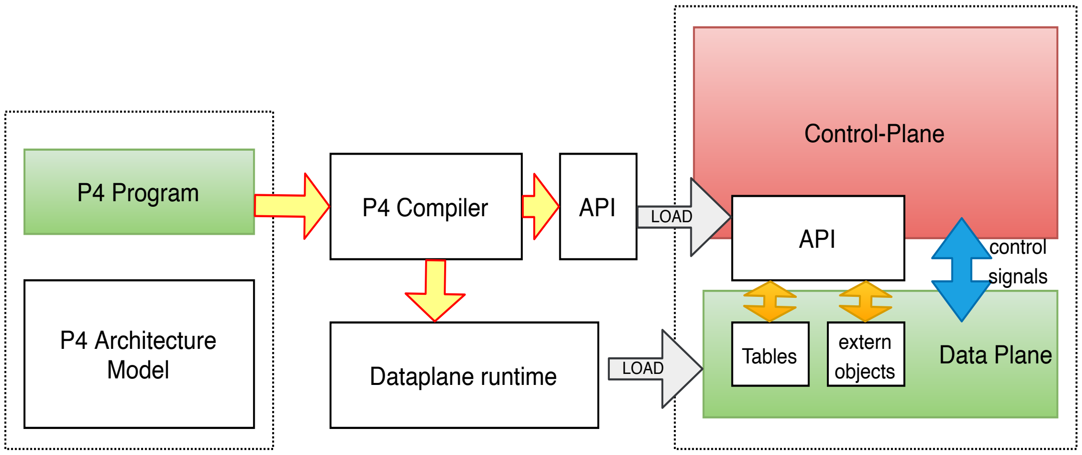

Figure 6: P4 Architecture (Source: Adapted from [64])

In a typical PISA switch, the execution of a P4 program is explained in Figure 7 (Hang et all.) which is summarized as the following steps:

1) The user develops a P4 program, which can be any type of network function, such as router, firewall, load balancer or packet inspection switch.
2) P4 compiler compiles the program as a JSON file and sends it to the switch, which can be a physical switch or a software model of it.
3) The states of parser, match-parser, match-action tables, ingress, egress queue and deparser are controlled by P4 execution.
4) The states of match-action tables are additionally controlled by control-plane with can change the behavior of the P4 code at runtime.

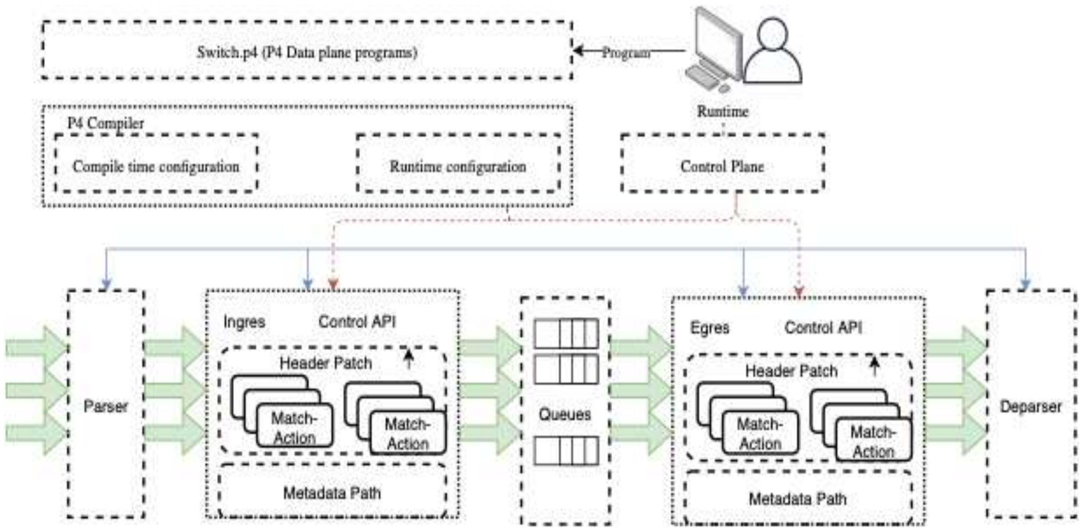

Figure 7: Pipeline execution in a P4-enabled switch (Hang et al., 2019)

P4 programs ease the development of a network equipment code to a level that only 128 lines are enough to build a simple IP switch with header validation [65]. Although the language itself is simple, there are other tools that emit P4 language code from another high-level language, such as the work done by the authors [66], P4HDL, which generates P4 code from a pseudo-code.

2.10.3. In-band Telemetry with Programmable Switches

The above three requirements to develop a programable hardware are not the only features addressed by P4. One of the most promising features of P4 arises in the telemetry. In-band Network Telemetry (INT) is defined in P4 language as one of the main applications [67]. Since P4 executes at the packet-processing level, it can rewrite every segment of the packet header, including the custom headers. This type of modification cannot be done in traditional statically programmed hardware-based network equipment. P4 helps set up a data plane by using the packet headers appropriately to collect even more information on the network's status than what we can determine using conventional methods [68].

The idea behind INT is to collect telemetry metadata for each packet, including routing paths for the packet, entry and exit timestamps, the packets' latency, queue occupancy in a given node, use of egress port connections, and alike. These measurements can be produced by each network node and sent in the form of a report to the monitoring system. Another way to embed them in packets is to update them into allocated nodes at any node on the packet visits and connect them to the monitoring system. In a recent study, researchers used P4 INT experimental validation for telemetry-based monitoring applications on the multi-layer optical network switches [69]. Using the telemetry data and the integrated software around it, semi-automatic congestion control over optical network switches can be achieved with the currently available SDN/NFV systems.

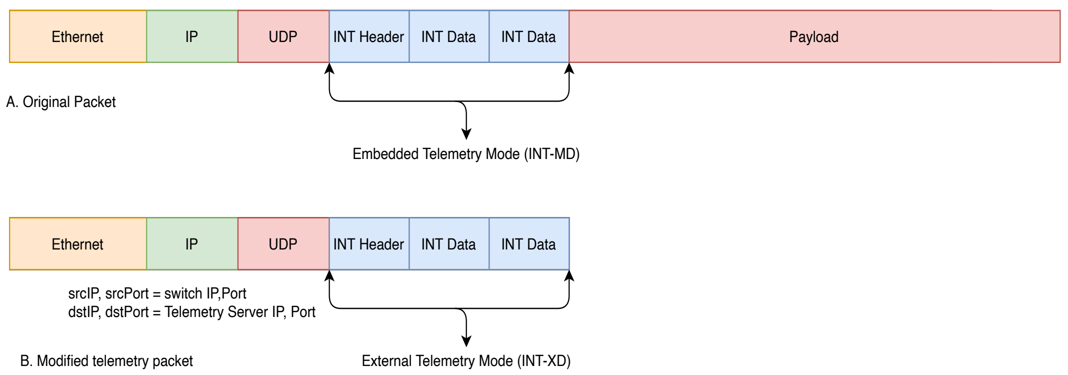
Ethernet
IP
UDP
INT Header
INT Data
INT Data
Payload
A. Original Packet
Embedded Telemetry Mode (INT-MD)
Ethernet
IP
UDP
INT Header
INT Data
INT Data
srcIP, srcPort = switch IP,Port
dstIP, dstPort = Telemetry Server IP, Port
B. Modified telemetry packet
External Telemetry Mode (INT-XD)

Figure 8: In-band Network Telemetry

Although telemetry data can be collected in any way that is defined by P4 code, there are two types of telemetry that are defined in a standard P4 implementation [70]. As shown in Figure 8, telemetry data can be either embedded within a packet, which is called INT-MD, or extracted as a separate packet, called INT-XD. INT-MD is usually used by intermediate routers (switches to identify any type of problem that might occur along the path, which INT-XD is useful for external applications that don't need the payload of the original packet.

2.11. Real-time Data Streaming

Real-time data streaming is shown to be beneficial for safety-critical networks by removing possible bottleneck situations at the data cumulation joints, such as the data aggregator switches at the industrial networks. In these networks, a possible delay in data would cause disastrous events, and data-streaming is a very good candidate solution as a remedy to this [71]. In the context of programmable switches, real-time data streaming is combined with telemetry in order to add application analytics, visibility, and troubleshooting features to a network stream. Apache Spark [72] and Apache Flink [73] are two of the most prominent software that is being used in streaming network telemetry data.

2.12. Deep Packet Inspection (DPI) and Application Layer Visibility

Deep Packet Inspection is important for telecommunication operators to gain more insight about the network and subscribers for revenue generation as well as cyber-security. A series of research [74], [75], [76] made in this area by the same author showed that subscriber profiling based on application-level classification is critical for operators to increase the revenue and generate insight about the network. As the name implies, DPI inspects every packet with respect to the source, destination, header information, payload, and any other layer that is wrapped into it. Application layer visibility enables operators to distinguish between their subscribers and offer them new subscription services accordingly. As the video content is on the rise, operators can offer subscribers based on their use of online video services, such as Netflix, Amazon Prime, or Hulu. In addition, DPI is a supportive tool in employing Lawful Intercept or applying some appropriate filters to the Internet access of children.

2.13. Future Directions and Challenges in Telecommunication Networks

Multiple transformations happening around us, the shift from VMs to container-based virtualization, improve the isolation, performance, and ease of operation, and bring a higher level of operations. The transformation in networking moves from legacy networking concepts into SDN-NFV based flexible, dynamic, agile, and more straightforward operation. The orchestrators are bringing dynamic resource management, service provisioning, and yet to come zero-touch networking and service management. The mind map Figure 9 gives a brief explanation of the continuous transformation of the telecommunication systems.

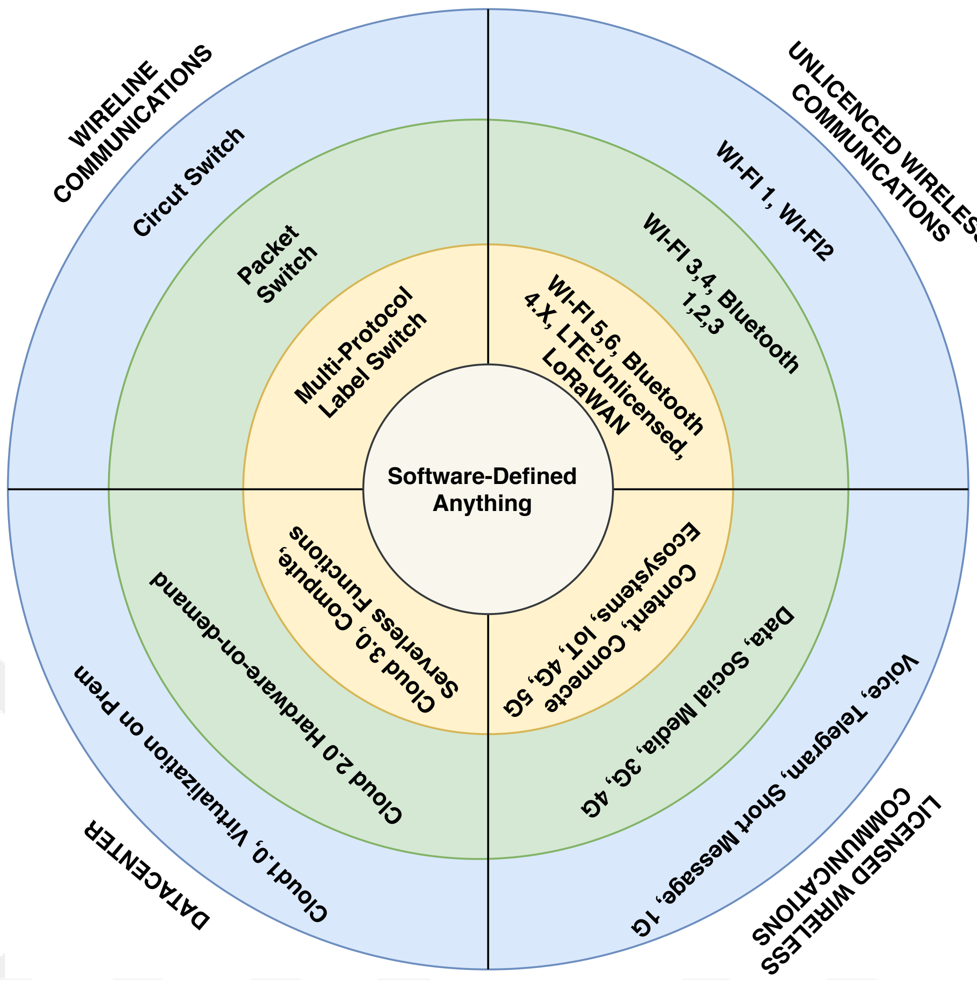
WIRELINE COMMUNICATIONS
Circuit Switch
Packet Switch
Multi-Protocol Label Switch
UNLICENCED WIRELESS COMMUNICATIONS
WI-FI 1, WI-FI2
WI-FI 3,4, Bluetooth 1,2,3
WI-FI 5,6, Bluetooth 4.X, LTE-Unlicensed, LoRaWAN
Software-Defined Anything
DATACENTER
Cloud1.0, Virtualization on Prem
Cloud 2.0 Hardware-on-demand
Cloud 3.0, Compute, Serverless Functions
LICENSED WIRELESS COMMUNICATIONS
Voice, Telegram, Short Message, 1G
Data, Social Media, 3G, 4G
Content, Connected Ecosystems, IoT, 4G, 5G

Figure 9: Mind map for the Transformation of the Networks

5G is a different business enabler for operators. It has different economics. It brings "programmable multi-access to the edge." This is an evolution in the access technologies where the operators are touching the customers through the Edge. Although 5G is promised to everyone by operators and governments, it is more important to start the services of 5G through any spectrum (4G) regardless of the spectrum. This allows the technology developers and operators to test their service while allowing the subscribers to see what is being promised in a nutshell.

One of the obvious things that are seen so far, because of the operators' low falling incomes, rather than competing in infrastructure installation, the countries will start to aggregate the infrastructures for active sharing among multiple operators. This active sharing will be more comfortable with the evolution of new technologies in the area allowing global telecommunication operators. There will be several operators delivering worldwide services by using automated systems. ONAP or similar network automation platforms will allow big players to access national infrastructures (legally) more easily.

In the past, any startup or a big vendor was going into the market with their strong abilities in hardware design and delivery in the telecommunication world. However, this becomes useless while the transformation steps up. With software-defined networking, the performance of the computing systems will be determined by the communication protocols since it will limit the performance of the feature, rather than HW and SW.

The above changes result from the evolution of the technologies. During the early times, we were talking about Information technologies. Later, this turned out to be on communication technologies. The future will be for Data Technologies. This convergence of the technologies shifted the focus of operators on Data Technologies to use AI and ML technologies effectively to increase the monetization from data processing such as subscriber data processing, capacity management, planning, or churn management.

Many projects are targeting the effect of AI and ML on telecommunication systems. While long-term tasks deal with availability and effectiveness, including network and operation planning and resource optimization in a planning perspective, short and medium-term tasks are related to real-time link scheduling, load balancing, and QoS (Quality of Service)/QoE (Quality of Experience) in the telecommunication world. In other words, while long-term tasks are more related to the management plane, medium and short-term tasks are more related to control and data planes.

In addition to the demands of the edge computing requirements, the domination of the mobile network (similar to the wireline) with video and other time-sensitive applications are already forcing the operators to use fiber optic cables as the connection medium of the base stations. Besides, higher capacity demands and time sensitivity in applications will shorten the range and reduce the base stations' capacity, and the number of concurrent subscribers served through a base station. While everyone is talking about 5G and the evolution of mobile networks, it will increase the use of fiber optic cable penetration throughout the entire world. Conversely, going towards 5G will improve the profitability of wireline operators in contrast to their loss expectations.

Networks and IT are converging; hence, the operation and planning teams shall also be planned accordingly. The new wave and inevitable trends will bring new standards for the telecommunication world to design the future as: i) Efficient Information Model for Data Collection, ii) Unified Flexible Interfaces, iii) Autonomous upgrade, and iv) Intend Driven Functionality Orchestration. Those aims require well-designed architectures, interface specifications and agile development and open standards, even open-source systems.

One of the main problems among telecommunication orchestration systems is the current developer who does not understand a carrier network's complexity. In the future, the operators will have software developers with extensive knowledge and experience in developing network applications, while the operation engineers having an in-depth understanding of networking. This trend will continue until the network automation and orchestration platforms reach their maturity level. Once they reach their maturity level, they will have built-in zero-touch networking and service management module which is already in a primitive era in itself. The rise of ZSM will reduce the need for any kind of operation staff in telecommunication world.

Edge computing is one of the biggest differentiators for communication service providers than cloud operators or OTTs. It is built around the only point where they physically touch their customers while no one else can. This makes the Edge a unique differentiator for communication service providers and uses it as a critical component of their 5G and IoT strategy in the next area of innovation for building new business opportunities.

CHAPTER 3

APPLICATION LAYER PROCESSING WITH P4 SWITCHES

3.1. Introduction

The transformation from legacy systems into software-defined architectures triggered the change in the hardware architectures. The demand for the change resulted in the development of PISA switches. The current state of the art in a PISA switch can scale up to 12.8 Tbps with a single ASIC/FPGA interface running with the speed of 400 Gbps. After the introduction of PISA switches in production environment, the applications running in L4, such as Load Balancers, Volumetric DDoS attack detection, and prevention systems, port-based DNS applications are being ported into PISA switches.

In this study, we aim to extend the use of PISA switches into L7 applications by designing a proper architecture. In the proposed architecture, by using PISA switches and its primary programming language, P4, an application-level traffic analyzing system is proposed in a software-based emulation environment. It's basically combining L4 analytics of P4 architecture and L7 properties of the current state of the art in DPI or similar application layer packet processing systems. The proposed architecture can be used to build a brand-new NGFW or DPI, by eliminating the complexities arising from switch-dependent code.

3.2. Current State of the Art (SOTA)

As of my knowledge, in the literature, the current SOTA in Programmable Switches consists of P4-based match-forward telemetry applications, stream processors and combination of these two techniques. Although there are many studies using P4 switches with such applications, the following applications are the most popular ones in their category.

3.2.1. Marple

Marple [77] is the first query language based on P4 language, in order to express packet matching tasks in a high-definition language, using functional constructs like filter, map, group by and zip. Marple targets PISA architecture software-only switches like BMV2 while also providing a simulation environment for switch pipeline. It aims to utilize the switch resources at minimum (i.e. memory and CPU) while providing streaming analytics capabilities at the switch level. Marple focuses mainly on packet-level In-band Telemetry, aiming to solve issues like delay, jitter, TCP in-cast and load imbalance across network links. The motivation behind Marple is to allow changing needs of an operator, enable to express there are of interest in network-related problems without having to redesign of hardware to each different monitoring task.

3.2.2. SONATA

SONATA [78] is a query-driven network telemetry system, based on P4 architecture and stream processor, deals with scalability issue by filtering the packets from the beginning before sending to stream processor for further operation. This approach comes with a trade-off between memory optimization in the data plane and losing flexibility in stream processor. Sonata tries to solve this problem by utilizing a query partitioning method that splits the query into two parts: Data plane and stream processor. While stream processor queries consist of simple constructs such as sum, count and join, data plane operations include every metadata related extraction in L4 properties of a packet. The focus is given to the data plane part, so that the query processing in stream processor decreases arbitrarily compared to classical methods, such as sending directly to stream processor.

3.2.3. Packetscope

PacketScope [79] is based on SONATA, which is a network telemetry system that lets to peek inside network switches to ask a suite of useful queries about how switches modify, drop, delay, and forward packets. It tries to eliminate the need for stream processor capabilities, mimic the operations of stream processor in a resource-limited environment, i.e., the switch itself. PacketScope expresses the queries by using tuples, converts the stream processor queries to tuple-based operations, and tags the packets as early as possible in the packet flow table of the P4 code, so that any packet-related metadata can be queried in a stateful way without the need for a stream processor. With this approach, PacketScope is useful in terms of detecting packet loss or latency in a PISA-based networking environment, such as detecting the queuing loss as the authors explain in their paper [79]. While PacketScope provides greater insight into the packet flow within a switch, it does not have any vision or focuses on the inspection of packets with respect to application-based classification.

3.2.4. Deep Match

Deep Match [80] is a novel approach to exploit the packet inspection properties of a PISA switch. The authors used P4 language to apply regular expression matches of the Redis [81] packet payload and select the routing accordingly. While it’s one of the first P4-based application layer inspection methods, this approach is limited to only one specific type of P4-based network interface card. They did not generalize it for any type of PISA switch and they only inspect the payload without combining the telemetry headers.

3.3. Proposed System Architecture

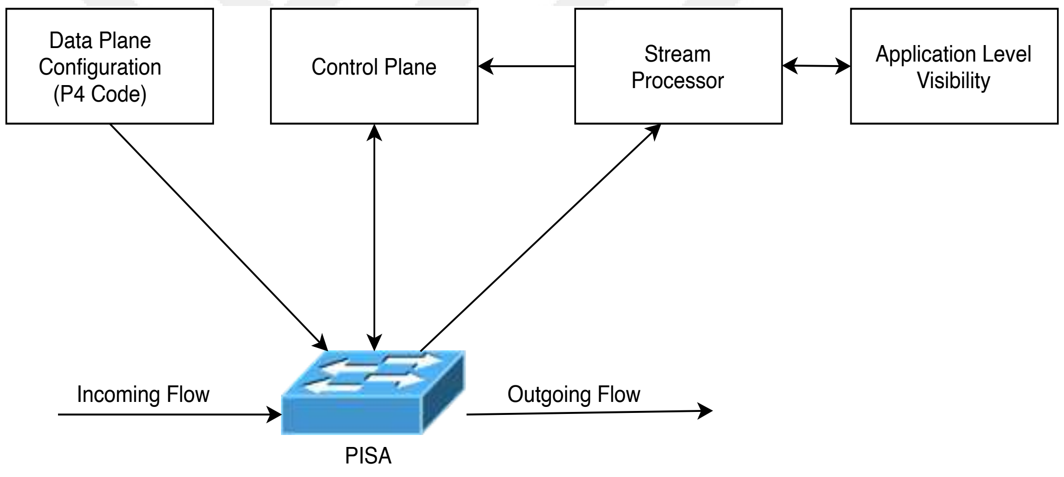
Data Plane Configuration (P4 Code)
Control Plane
Stream Processor
Application Level Visibility
Incoming Flow
Outgoing Flow
PISA

Figure 10: Proposed System Architecture

The proposed system architecture in Figure 10 consists of 5 main components: PISA, Stream Processor, Control Plane, and Data Plane Configuration.

PISA: Programmable Switch that can run multiple instances of different P4 code.

Data Plane: The generated P4 code for specific monitoring/telemetry/DPI/NGFW tasks. These P4 programs can be deployed according to specific task needs.

Control Plane: Programmable Switch related control plane engine to be placed. The control plane is aware of Data plane drivers, can communicate with the underlying switch according to the specific tasks. Although the proposed architecture supports any application-specific task, from now on the architecture will be coupled with DPI use case to make it easier to understand. This module is DPI-aware, which is fed from the specific packet stream so that any decision to be made on the switch can be controlled by examining the specific packets.

Stream Processor: The stream processor operates on the matching stream patterns based on the decisions taken from data plane configuration. Specific telemetry tasks can be offloaded to stream processor in order to decrease the workload over the switch or vice versa. Workload trade-off between the stream processor and the switch is based on the number of streams that match a specific monitoring task.

Application-Level Visibility and Control: Application-level visibility and control is the component that actually identifies the types of applications based on their L4 to L7 properties, which is also called DPI.

In a typical DPI system, a server with network interfaces is running the DPI application. There are two usage modes of DPI systems which are active and passive DPI systems. In the passive mode, they are fed by mirror of the traffic and processes offline. On the other hand, active DPI systems fall within the whole traffic and are supposed to process all the traffic piece by piece in real-time

In the proposed architecture, the PISA switch processes the packets in the network layer, even can process the flows in the transport layer and co-operates with the stream processor to identify the applications. This is the point where the aggregation-disaggregation of high-performing PISA switch and application identification engines.

The PISA switch selects the minimal packets from the flows and forwards them to the stream processor/DPI engine to identify the applications and generate the actions among the predefined policies. The proposed architecture combines the power of PISA and L7 application inspection/classification/processing/control features by designing them together. The simulation results indicate that in the near future most of the systems using application awareness will re-design their systems running on top of PISA switches together with their redesigned applications as a stream processor.
The following algorithm explains our approach:

```
While packet -> in ingres buffer
      Extract telemetry headers
      Put in Flow-Keys Telemetry Headers
      If Flow Not in Flow-Table
            Create flow in Flow-Table
      Else IF Flow-Packet-Count < 2
```

```
            Put Payload in Flow-Packets with Flow-Keys in Flow-
Table
            Continue
      Else
            Create telemetry header with INT-XD options
            Send Flow-Table in Flow-Keys to External Telemetry
```

The accurate accounting of the flows can also be done with P4 language.

The accounting of a flow should include the following information:

Considering the definition of the flow,

```
For every flow,
      count
            number of packets,
            number of bytes,
            flow start time,
            flow end time,
in addition to that,
      for TCP flows, TCP flags.
```

The P4 code on switch would combine the accounting information and send the rest to the aggregator with following pseudo-code:

```
// Flow key registers
reg_src_ip = Register();
reg_dst_ip = Register();
reg_proto  = Register();
```

```
reg_l4  = Register();


// Flow statistics registers
reg_pkt_count = Register();
reg_byte_count = Register();
reg_time_start = Register();
reg_time_end = Register();
reg_flags = Register();
initialize_registers(hdr: PacketHeader, index: HashIndex, md: Metadata):
reg_src_ip[index] = hdr.src_ip;
reg_dst_ip[index] = hdr.dst_ip;
reg_proto[index] = hdr.proto;
reg_l4[index] = hdr.l4;
reg_pkt_count[index] = 1;
reg_byte_count[index] = length(hdr.ethernet) + hdr.ip_len
reg_time_start[index] = md.timestamp;
reg_time_end[index] = md.timestamp;
reg_flags[index] = hdr.tcp_flags;
with pkt = ingress.next_packet():
hdr = parse(pkt);
md = pkt.metadata;
index = hash({hdr.src_ip, hdr.dst_ip, hdr.proto, hdr.l4});
collision = hdr.src_ip != reg_src_ip[index]
|| hdr.dst_ip != reg_dst_ip[index]
|| hdr.proto
!= reg_proto[index]
|| hdr.l4
!= reg_l4[index]
if collision:
```

```
// Export info and keep track of new flow
flow_record = { reg_src_ip[index],
reg_dst_ip[index],
reg_proto[index],
reg_l4[index],
reg_pkt_count[index],
reg_byte_count[index],
reg_time_start[index],
reg_time_end[index],
reg_flags[index] }
emit({hdr.ethernet, flow_record});
initialize_registers(hdr, index, md);
else:
// Update statistics of current flow
reg_pkt_count[index] += 1;
reg_byte_count[index] += length(hdr.ethernet) + hdr.ip_len
reg_time_end[index] = md.timestamp;
reg_flags[index] ||= hdr.tcp_flags;
```

This pseudo-code works as the preprocessor of the flow, extracts the required fields and sends them to application layer stream processor for further processing.

Lastly, the traditional DPI systems have two operating modes:

- Inline
- Out-of-Band

In the inline mode, DPI systems are placed between the edge and core network, so that the traffic is processed as the flow continues. This operating mode enables DPI to apply policies directly on the flow without requiring any other hardware. The biggest disadvantage of this approach is that the DPI becomes the weakest link of the network, it should be scaled at least as much as the aggregated sum of the traffic received from the edges.

In Out-of-band mode, DPI acts like a simple traffic analyzing tool, it received the traffic passively from a mirror port of a network aggregation device, collecting all the traffic information and applying policies accordingly. In this mode, the biggest challenge is policy application, as the traffic is not directly passing through the DPI, it can only act on TCP traffic by sending TCP-resets to the source addresses, for example, in order to apply a restricted access policy to a particular destination address within the scope of the network. Other types of policy applications, such as bandwidth restriction, quality-of-service changes etc., require control plane integration with the underlying network device.

Our architecture also combines the benefits of inline DPI devices with the out-of-band ones where the traffic is actively received on the switch, counted, and reported on the aggregated external devices and the policies are actively applied as the events triggers occur.

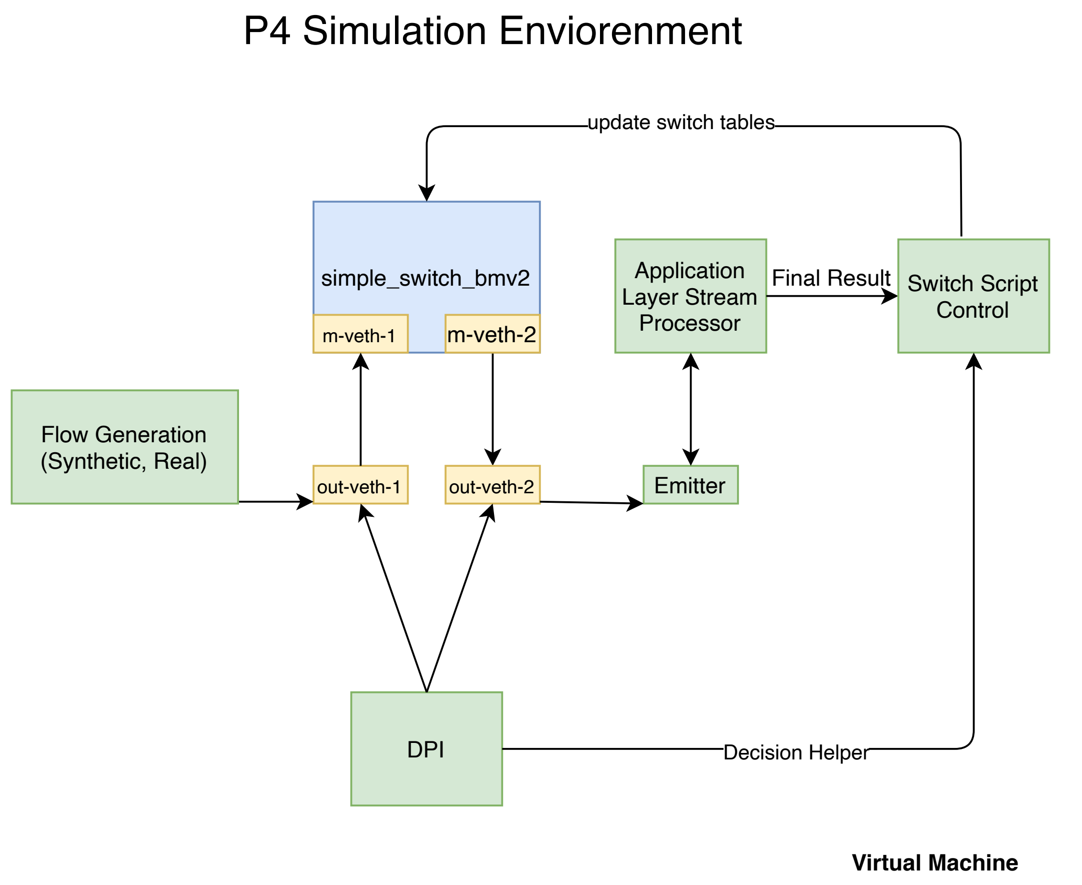

Figure 11: Simulation Environment

In order to simulate the proposed architecture, the following components are built as a development and simulation environment:

P4 Simulation Environment: This is the default simulation target for BMV2 PISA switches, as shown in Figure 11, which includes Mininet by default and handles virtual NIC creating, switch port allocation, connecting the switch port to host process, and running the rest of the packet flow.

Virtual Machine: This is the default virtual machine, build programmatically with Vagrant, developer friendly VM running environment based on Ubuntu 14.04 (ubuntu/trusty64) and several other necessary components.

Simple_switch_bmv2: BMV2 software switch, based on Python2.7

```
m−veth−1   : Ingres mininet Switch Port
m−veth−2   : Egres mininet Switch Port
out−veth−1 : Ingrest Server Host Port
out−veth−2 : Egres Server Host Port
```

Flow Generation: This is the controlled flow generation tool, written in Go. Synthetic flows are created with Python, while real-flows are taken from Canadian Institute for Cyber-Security [82] .

DPI: Deep Packet Inspection module written in Go, based on nDPI [83].

Emitter: Flow emitter that reads from the mirroring port, extracts metadata header information written by Data-Plane and sends the rest of the packet for stream processor. This module is also Apache-Spark aware; the final result of the telemetry query is calculated by Emitter module.

Application Layer Stream Processor: The streaming processor for the rest of the flows that match the final criteria for the expected output. In this simulation, we used Apache-Spark as stream processor. The stream processor will be upgraded to Apache-Flink for better performance and scalability.

Switch Script Control: This script controls the switch tables in order to update the relevant switch tables under control.

3.5. Example Use Case: Running DDoS Attack Simulation

The simulation setup is as shown in Figure 12:

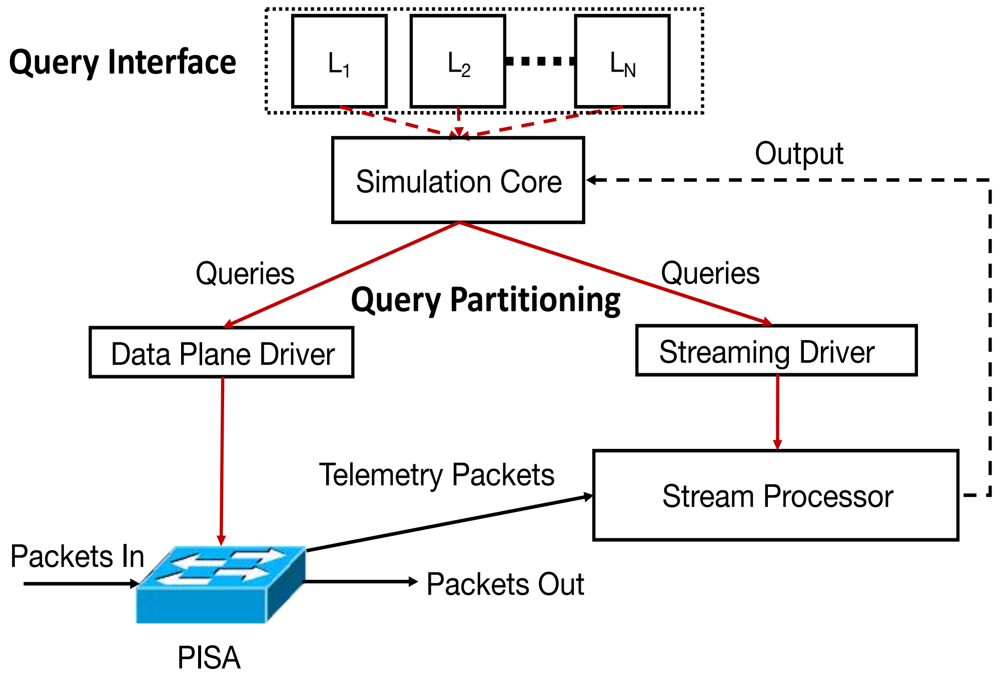

Figure 12: DDoS Attack Simulation Setup

Using our Python script, the following synthetic traffic is generated:

30 seconds of normal traffic from the start to end.
15 seconds of attack traffic after 5. Second till 20. second

50 packets of normal network traffic per second
(srcIP = random, srcPort=Linux_ephemeral, dstIP = random, dstPort=80 type=TCP)
400 packets of DDoS Traffic for dstIP = 99.7.186.25, dstPort = 53, srcIP = random, srcPort = Linux_ephemeral, type=UDP, DNS=ns-query)
Total number of packets = 30 x 50 + 400 x 15 = 7.500 packet
Threshold for DDoS Detection = 100 random srcIP hitting one dstIP for the entire duration of simulation.

Telemetry Query Decomposition

```
    ddos = (PacketStream(1)
 L1     .map(keys=('ipv4.dstIP', 'ipv4.srcIP'))
 L2     .distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
 L3     .map(keys=('ipv4.dstIP',), map_values=('count',),
func=('set', 1,))
 L4     .reduce(keys=('ipv4.dstIP',), func=('sum',))
 L5     .filter(filter_vals=('count',), func=('geq', T))
 L6     .map(keys=('ipv4.dstIP',))
        )

L1: extract dstIP, srcIP from the packet
L2: apply “distinct” on dstIP, srcIP pairs
L3: define “count” field for  each distinct dstIP, srcIP value pairs
L4: appt “sum” on on field “count” for dstIP value only
L5: select the dstIP, count > Threshold value pair
L6: write the result
```

The execution of this query is controlled by the last parameter of config array

```
    queries = [ddos]
    config["final_plan"] = [(1, 32, 5)]
```

Parameter 1: Query id (which is given in PacketStream())
Parameter 2: Query Level (1-32, 32 is the finest query level on the packet)
Parameter 3: Query Execution Level on switch (L1 = query is only executed on switch at first level, L5 = query is executed on switch)

Level - 1

Dataplane Queries:

```
for 10032
in
.map(keys=['count','ipv4.srcIP','ipv4.dstIP'],map_keys=(u'ipv4.dstIP
',), values=[], map_values=[], func=('mask', 32))
.map(keys=('ipv4.dstIP', 'ipv4.srcIP'), map_keys=[], values=[],
map_values=[], func=[])
```

Streaming Queries:

```
for 10032
in
      .map(lambda
((ipv4_dstIP,ipv4_srcIP)):((ipv4_dstIP,ipv4_srcIP)))
```

```
      .distinct()
      .map(lambda ((ipv4_dstIP,ipv4_srcIP)):((ipv4_dstIP),(1)))
      .reduceByKey(lambda x,y: x+y)
      .filter(lambda ((ipv4_dstIP),(count)):((float(count)>=100 )))
      .map(lambda ((ipv4_dstIP),(count)):((ipv4_dstIP)))
```

Level - 2

Data-plane Queries:

```
for 10032
in
      .Map(keys=['count', 'ipv4.srcIP', 'ipv4.dstIP'],
map_keys=(u'ipv4.dstIP',), values=[], map_values=[], func=('mask',
32))
      .Map(keys=('ipv4.dstIP', 'ipv4.srcIP'), map_keys=[],
values=[], map_values=[], func=[])
      .Distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
```

Streaming Queries:

```
for 10032
in
      .map(lambda ((ipv4_dstIP,ipv4_srcIP)):
((ipv4_dstIP,ipv4_srcIP)))
      .map(lambda ((ipv4_dstIP,ipv4_srcIP)): ((ipv4_dstIP),(1)))
      .reduceByKey(lambda x,y: x+y)
      .filter(lambda ((ipv4_dstIP),(count)): ((float(count)>=100 )))
      .map(lambda ((ipv4_dstIP),(count)): ((ipv4_dstIP)))
```

Level - 3

Data-plane Queries:

```
for 10032
in
      .Map(keys=['count', 'ipv4.srcIP', 'ipv4.dstIP'],
map_keys=(u'ipv4.dstIP',), values=[], map_values=[], func=('mask',
32))
      .Map(keys=('ipv4.dstIP', 'ipv4.srcIP'), map_keys=[],
values=[], map_values=[], func=[])
      .Distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
      .Map(keys=('ipv4.dstIP',), map_keys=[], values=[],
map_values=['count'], func=('set', 1))
```

Streaming Queries:

```
For 10032
in
      .map(lambda                               ((ipv4_dstIP,count)):
((ipv4_dstIP),(float(count))))
      .reduceByKey(lambda x,y: x+y)
      .filter(lambda ((ipv4_dstIP),(count)): ((float(count)>=100 )))
      .map(lambda ((ipv4_dstIP),(count)): ((ipv4_dstIP)))
```

Level - 4

Data-plane Queries:

```
for 10032
in
      .Map(keys=['count', 'ipv4.srcIP', 'ipv4.dstIP'],
map_keys=(u'ipv4.dstIP',), values=[], map_values=[], func=('mask',
32))
      .Map(keys=('ipv4.dstIP', 'ipv4.srcIP'), map_keys=[],
values=[], map_values=[], func=[])
      .Distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
      .Map(keys=('ipv4.dstIP',), map_keys=[], values=[],
map_values=['count'], func=('set', 1))
      .Reduce( keys=(ipv4.dstIP), values=(count), func=('sum',),
threshold=1)
      .Filter(prev_keys=('ipv4.dstIP',), filter_keys=[],
filter_vals=('count',), func=('geq', 100) src = 0)
```

Streaming Queries:

```
for 10032
in
      .map(lambda                               ((ipv4_dstIP,count)):
((ipv4_dstIP),(float(count))))
      .reduceByKey(lambda x,y: x+y)
      .filter(lambda ((ipv4_dstIP),(count)): ((float(count)>=100 )))
      .map(lambda ((ipv4_dstIP),(count)): ((ipv4_dstIP)))
```

Level - 5:

Data-plane Queries:

```
for 10032
in
      .Map(keys=['count', 'ipv4.srcIP', 'ipv4.dstIP'],
map_keys=(u'ipv4.dstIP',), values=[], map_values=[], func=('mask',
32))
```

```
      .Map(keys=('ipv4.dstIP', 'ipv4.srcIP'), map_keys=[], 
values=[], map_values=[], func=[])
      .Distinct(keys=('ipv4.dstIP', 'ipv4.srcIP'))
      .Map(keys=('ipv4.dstIP',), map_keys=[], values=[], 
map_values=['count'], func=('set', 1))
      .Reduce( keys=(ipv4.dstIP), values=(count), func=('sum',), 
threshold=1)
      .Filter(prev_keys=('ipv4.dstIP',), filter_keys=[], 
filter_vals=('count',), func=('geq', 100) src = 0)
```

Streaming Queries:

```
for 10032
in
      .map(lambda ((ipv4_dstIP,count)): 
((ipv4_dstIP),(float(count))))
      .map(lambda ((ipv4_dstIP),(count)): ((ipv4_dstIP)))
```

Using the telemetry data, the following reduction in Table 2 is achieved:

Table 2 Reduction of Packets based on Telemetry Levels

Telemetry Level	**Outgoing Packets**	**Incoming Packets**
Level 1	7480	7500
Level 2	7018	7500
Level 3	5879	7500
Level 4	4	7500
Level 5	4	7500

Level 1: standard packet forwarding switch

Level 2: distinct srcIP, dstIP

Level 3: report distinct src, dst ip address list, and increment dstIp count by 1

Level 4: report distinct src, dst ip address list, and increment dstIp count by 1 and sum

Level 5: report distinct src, dst ip address list, and increment dstIp count by 1 and sum and apply threshold value

CHAPTER 4

EXPERIMENTAL STUDY

4.1. Experiment-1: Application Identification Performance Improvement DPI

Application Classification on Mixed flow captures

Our hypothesis is that in order to identify an application in a packet, few bytes in a flow should be enough to determine the type of application correctly [84], [85]. Keeping this in mind, we must first identify the session in a packet. This use case demonstrates the performance improvement in DPI systems by eliminating the number of packets by some factor.

Session Identification in an IP flow is based on two different IP sessions:

a. TCP Session

SrcIP, DstI, SrcPort, DstPort, TCPSeqNum

TCP Session Identification is based Source IP, Destination IP, Source Port Destination Port and the TCP Sequence Number. The TCP session is established after the 3-way handshake:

```
Source -> Destination (SYN+Seq #)
Destionation -> Source (SYN ACK+Seq #')
Source -> Destination (ACK+Seq #'')
```

After the last ACK of the source, Sequence Number is incremented for a flow in the TCP session. Actually, it comes from the nature of TCP. It starts randomly and increments by the amount of the data transferred in each packet. Same is valid for ACK number.

The packets that will be reduced should be the packets after this 3-way handshake packets. In order to identify the flows, we will use the packet SYN `ACK,` and the response to the third packet. In other words, the first two packets of the server (or destination to source will be kept).

b. UDP Session

SrcIP, DstI, SrcPort, DstPort,

UDP is a connectionless protocol; there is no clear definition of a UDP session. Every packet may create a flow independently. Basic identification for UDP flow consists of Source IP, Destination IP, Source Port, and Destination Port. Since Source Port is randomly allocated depending on the OS (which is called ephemeral ports), any flow that is using the same source port is considered as the same UDP session.

4.2. Sample Packet Captures

In order to study the flow reduction, we used the sample captures from nDPI that is used for verification of protocol identification. The capture files consist of 183 files, containing more than one protocol in one capture file. 22 files that are too small for reduction (having packets less than 2) are excluded from the study. 1 packet especially crafted for testing invalid packet type is also excluded since we are interested in valid packets, leaving us 160 packet captures.

In order to reduce the flow following pseudo-code is used:

```
network_packets = rdpcap(infile)
sessions = network_packets.sessions()
```

```
for key in sessions:
        pktCount=0
        for pkt in sessions[key]:
                if (pktCount < 2):
                        write(pkt, outfile)
                        pktCount = pktCount + 1
```

In this code, sessions are extracted by the criteria, whether they are TCP or UDP session. As mentioned earlier, for TCP sessions, 3-way handshake packets are excluded from the session, whereas, for UDP sessions, there is no precondition to exclude the packets. We use the 2nd packet of the 3-way handshake as the first packet of the flow. We use the first packet after SYN ACK packet from server to the client as the first packet of the flow.

After the extraction of sessions, nDPI sample classifier is used to classify the application in each reduced capture by replaying the capture file on the switch.
A sample for OpenVPN is given below.

Table 3 Results for OpenVPN Traffic Reduction

	Traffic Statistics for OpenVPN	
	Original	**Reduced**
Ethernet bytes	64263	1392
Discarded bytes	0	0
IP packets	298	12
IP bytes	57111	1104
Unique flows	3	3
TCP Packets	95	4
UDP Packets	203	8
Max Packet size	1480	162
Packet Len < 64	98	11
Packet Len 64–128	73	0
Packet Len 128–256	101	1
Packet Len 256–1024	17	0
Packet Len 1024–1500	9	0
Packet Len > 1500	0	0
nDPI throughput	45.88 Kpps/75.49 Mb/sec	41.38Kpps/36.62 Mb/sec
Analysis begin	07/Jul/2016 18:22:26	07/Jul/2016 18:22:26
Analysis end	28/Aug/2016 00:55:09	28/Aug/2016 00:54:52
Traffic throughput	0.00 pps / 0 b/sec	0.00 pps / 0 b/sec
Traffic duration	4429962.500 sec	4429946.000 sec
OpenVPN Packets	298	4
OpenVPN Bytes	57111	356

The following tables show the results of the experiment.

Table 4 Rates for Test Captures

μ REDUCTION RATIO	82%
μ REDUCTION FACTOR	5.5
μ DETECTION RATE	84%

Table 5 Fully Detected Protocols from the capture files

	BYTES		PACKETS		DETECTION		REDUCE
	ORG	RDC	ORG	RD	POS.	NEG.	
anydesk	2962572	767	6963	8	1	0	99,97
exe_download	734335	328	703	4	1	0	99,96
exe_download_as	542265	328	534	4	1	0	99,94
tor	3106096	3524	3859	42	4	0	99,89
whatsappfiles	467113	760	620	8	1	0	99,84
wireguard	791758	1576	2399	4	1	0	99,80
ps_vue	2242710	5184	1740	15	3	0	99,77
tls_long_cert	121969	380	182	4	1	0	99,69
ftp	1158196	3805	1192	12	3	0	99,67
quic-mvfst	408962	1414	353	2	1	0	99,65
git	76165	376	90	4	1	0	99,51
netflix	6323017	32776	6999	217	5	0	99,48
coap_mqtt	954917	5505	8516	51	3	0	99,42
dns-tunnel	80668	528	438	8	1	0	99,35
bitcoin	596362	4816	637	24	1	0	99,19
wa_video	998593	8587	1567	38	6	0	99,14
ssh	41738	401	258	4	1	0	99,04
quic_t51	589126	5664	642	4	1	0	99,04
quic-28	252865	2782	253	4	1	0	98,90
bittorrent_ip	519514	6512	479	8	1	0	98,75
skype-conf	44487	616	200	4	1	0	98,62
dns_exfiltr	80745	1149	300	4	1	0	98,58
instagram	3009247	47580	3443	122	7	0	98,42
tls_verylong_ce	23381	380	48	4	1	0	98,37
check_mk_new	22594	391	98	4	1	0	98,27
quic-mvfst-22	300063	5232	490	4	1	0	98,26
bad-dns-traffic	108542	1934	382	12	1	0	98,22
capwap	108037	2113	422	21	2	0	98,04
anyconnect-vpn	1088929	23234	3001	166	17	0	97,87
openvpn	64263	1392	298	12	1	0	97,83
webex	902823	19937	1580	223	6	0	97,79
bittorrent_utp	43553	979	86	4	1	0	97,75
facebook	31951	752	60	8	1	0	97,65
nintendo	357057	9156	1000	66	3	0	97,44
simple-dnscrypt	47340	1344	111	16	1	0	97,16
443-opvn	12677	380	46	4	1	0	97,00
Oscar	11090	352	71	4	1	0	96,83
google_ssl	9780	328	28	4	1	0	96,65
nest_log_sink	137036	4806	1000	60	3	0	96,49
modbus	9129	358	102	4	1	0	96,08
quic046	93697	3723	100	4	1	0	96,03
fix	145778	5858	1261	48	1	0	95,98
weibo	279507	11287	498	104	6	0	95,96
tls_esni_sni_b	16811	696	38	8	1	0	95,86
pps	2307979	104799	2557	243	4	0	95,46

http-crash-	3544	168	9	2	1	0	95,26
smb_deletefile	33172	1660	101	4	1	0	95,00
WebattackXSS	4946124	248266	9374	2641	1	0	94,98
teams	1554287	78248	2817	267	15	0	94,97
dnp3	51786	2752	543	32	1	0	94,69
wechat	707438	43775	1672	287	15	0	93,81
s7comm	6580	408	55	4	1	0	93,80
telegram	374409	25197	1566	119	15	0	93,27
youtube_quic	198575	13389	289	12	2	0	93,26
1kxun	664361	45690	1439	297	16	0	93,12
bittorrent	312904	21595	299	74	1	0	93,10
ja3_lots_of1	7614	528	27	4	1	0	93,07
ja3_lots_of2	5396	380	11	4	1	0	92,96
wa_voice	187832	13276	736	76	11	0	92,93
viber	157311	12098	424	81	9	0	92,31
youtubeupload	130326	10358	137	12	1	0	92,05
dropbox	110884	9056	848	48	1	0	91,83
amqp	27354	2284	160	12	1	0	91,65
iphone	232616	21922	500	138	12	0	90,58
skype	708140	71068	3284	639	13	0	89,96
WebattackSQLinj	32264	3384	94	36	1	0	89,51
quic	360998	37893	518	34	4	0	89,50
hangout	3230	340	19	2	1	0	89,47
ssdp-m-search	1653	174	19	2	1	0	89,47
BGP_Cisco_hdlc	1305	144	14	2	1	0	88,97
dos_win98_smb_	10055	1130	220	9	3	0	88,76
skype_unknown	537720	60508	2146	537	13	0	88,75
netbios	30922	3546	260	24	2	0	88,53
sip	51847	5966	112	11	3	0	88,49
whatsapp_l_call	223130	26502	1253	187	11	0	88,12
rx	29643	3641	132	18	1	0	87,72
6in4tunnel	43341	5326	127	26	5	0	87,71
android	143354	18809	500	167	14	0	86,88
ajp	7414	1020	38	10	2	0	86,24
quic_q46	21721	3028	20	4	1	0	86,06
quic_q50	20914	3048	20	4	1	0	85,43
ethereum	264111	39317	2000	260	2	0	85,11
malware	8625	1347	26	10	4	0	84,38
teamspeak3	2223	354	13	2	1	0	84,08
quic_q39	25625	4131	60	4	1	0	83,88
iec60780-5-104	12561	2034	147	24	1	0	83,81
whatsapp_login	32369	5963	93	19	7	0	81,58
whatsapp_voice_	34319	6492	261	52	3	0	81,08
quic-mvfst-exp	27029	5272	30	4	1	0	80,50
netflowv9	14128	2832	10	2	1	0	79,95
ftp_failed	2132	476	18	4	1	0	77,67
smpp_in_general	1552	347	17	4	1	0	77,64
EAQ	26563	6732	197	82	2	0	74,66
upnp	10248	2928	14	4	1	0	71,43
fuzz-2020-02	158043	46445	366	125	3	0	70,61
quic-29	9746	3011	15	4	1	0	69,11
quic-24	8360	3029	15	4	1	0	63,77
zabbix	955	376	10	4	1	0	60,63
4in4tunnel	970	388	5	2	1	0	60,00
quic-27	13367	5664	20	4	1	0	57,63
quic-mvfst-27	13367	5664	20	4	1	0	57,63
quic_q46_b	7500	3239	20	4	1	0	56,81
fuzzing	32268	15422	131	81	3	0	52,21
mongodb	3388	1648	27	16	2	0	51,36

mssql_tds	17172	8728	38	20	1	0	49,17
malformed_dns	6004	3096	6	4	1	0	48,43
quic-23	7671	3956	20	4	1	0	48,43
fuzz-2006	99986	53930	691	399	9	0	46,06
dnscrypt-v2-doh	230431	132987	577	136	1	0	42,29
skype_udp	459	278	5	3	1	0	39,43
teredo	3150	1980	24	14	1	0	37,14
quic_t50	8708	5664	12	4	1	0	34,96
smbv1	1365	895	7	4	1	0	34,43
diameter	2124	1488	6	4	1	0	29,94
websocket	561	428	5	4	1	0	23,71
steam	11516	10218	104	97	1	0	11,27
kerberos	30139	29412	77	75	4	0	2,41
encrypted_sni	2382	2382	3	3	1	0	0,00
tls-esni-fuzzed	2382	2382	3	3	1	0	0,00
4in6tunnel	2284	2284	4	4	1	0	0,00
mysql-8	463	463	4	4	1	0	0,00
ubntac2	1928	1928	8	8	1	0	0,00
filtered	21595	21595	74	74	1	0	0,00
dnscrypt-v1	321274	321274	608	564	2	0	0,00
WebattackRCE	210131	210131	797	797	2	0	0,00

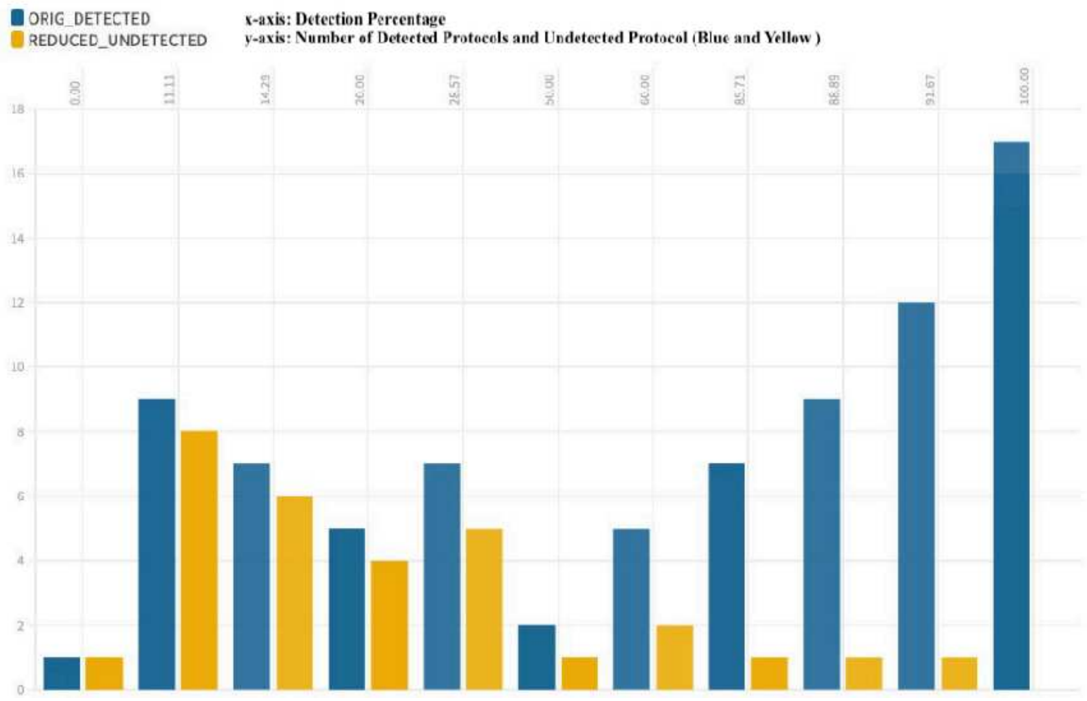

Figure 13: Detected vs. Undetected Applications in Reduced Flow

4.3. Experiment-2: TCP-based Application Identification

In the second experiment, we used the real captures from Canadian Institute for Cybersecurity [82], namely the files in the dataset named “PCAP−01−12_0750-0818”.

There are 69 files located in this dataset; each containing a real-world data capture that contains data from a real DDoS attack along with different types of traffic. In the data, there are multiple protocols in a flow which makes our life harder. In table 7, the identification percentage comes from having multiple different protocols in a single flow.

In order to see the effect of proposed method on TCP traffic, we extracted the TCP streams and used the extracted streams to send to the simulation.

Following results are achieved:

Table 6 Rates for Real-life Captures Using only TCP Streams

μ REDUCTION RATIO	97.88%
μ REDUCTION FACTOR	47.16
μ DETECTION RATE	95%

Table 7 TCP-based reduction results

FILE	A	B	C
SAT-01-12-2018_0750.pcap	2	2	100%
SAT-01-12-2018_0751.pcap	3	3	100%
SAT-01-12-2018_0752.pcap	3	3	100%
SAT-01-12-2018_0753.pcap	4	3	75%
SAT-01-12-2018_0754.pcap	4	4	100%
SAT-01-12-2018_0755.pcap	4	4	100%
SAT-01-12-2018_0756.pcap	4	4	100%

SAT-01-12-2018_0757.pcap	3	3	100%
SAT-01-12-2018_0758.pcap	1	1	100%
SAT-01-12-2018_0759.pcap	3	3	100%
SAT-01-12-2018_0760.pcap	5	4	80%
SAT-01-12-2018_0761.pcap	4	4	100%
SAT-01-12-2018_0762.pcap	4	4	100%
SAT-01-12-2018_0763.pcap	4	4	100%
SAT-01-12-2018_0764.pcap	4	4	100%
SAT-01-12-2018_0765.pcap	3	3	100%
SAT-01-12-2018_0766.pcap	3	3	100%
SAT-01-12-2018_0767.pcap	3	3	100%
SAT-01-12-2018_0768.pcap	4	4	100%
SAT-01-12-2018_0769.pcap	4	4	100%
SAT-01-12-2018_0770.pcap	4	4	100%
SAT-01-12-2018_0771.pcap	3	3	100%
SAT-01-12-2018_0772.pcap	4	3	75%
SAT-01-12-2018_0773.pcap	3	3	100%
SAT-01-12-2018_0774.pcap	1	1	100%
SAT-01-12-2018_0775.pcap	3	3	100%
SAT-01-12-2018_0776.pcap	4	4	100%
SAT-01-12-2018_0777.pcap	2	2	100%
SAT-01-12-2018_0778.pcap	5	5	100%
SAT-01-12-2018_0779.pcap	3	3	100%
SAT-01-12-2018_0780.pcap	4	4	100%
SAT-01-12-2018_0781.pcap	3	3	100%
SAT-01-12-2018_0782.pcap	5	4	80%
SAT-01-12-2018_0783.pcap	4	4	100%
SAT-01-12-2018_0784.pcap	5	5	100%
SAT-01-12-2018_0785.pcap	1	1	100%
SAT-01-12-2018_0786.pcap	4	3	75%
SAT-01-12-2018_0787.pcap	5	5	100%
SAT-01-12-2018_0788.pcap	7	6	86%
SAT-01-12-2018_0789.pcap	6	5	83%
SAT-01-12-2018_0790.pcap	5	5	100%
SAT-01-12-2018_0791.pcap	5	5	100%
SAT-01-12-2018_0792.pcap	6	5	83%
SAT-01-12-2018_0793.pcap	4	4	100%
SAT-01-12-2018_0794.pcap	5	4	80%
SAT-01-12-2018_0795.pcap	5	5	100%
SAT-01-12-2018_0796.pcap	5	5	100%
SAT-01-12-2018_0797.pcap	7	7	100%
SAT-01-12-2018_0798.pcap	5	5	100%
SAT-01-12-2018_0799.pcap	4	4	100%
SAT-01-12-2018_0800.pcap	5	5	100%
SAT-01-12-2018_0801.pcap	5	5	100%
SAT-01-12-2018_0802.pcap	4	4	100%
SAT-01-12-2018_0803.pcap	4	4	100%
SAT-01-12-2018_0804.pcap	4	4	100%
SAT-01-12-2018_0805.pcap	4	4	100%
SAT-01-12-2018_0806.pcap	3	3	100%
SAT-01-12-2018_0807.pcap	5	4	80%
SAT-01-12-2018_0808.pcap	6	5	83%
SAT-01-12-2018_0809.pcap	6	6	100%
SAT-01-12-2018_0810.pcap	5	5	100%
SAT-01-12-2018_0811.pcap	4	4	100%
SAT-01-12-2018_0812.pcap	3	3	100%
SAT-01-12-2018_0813.pcap	6	5	83%
SAT-01-12-2018_0814.pcap	7	6	86%
SAT-01-12-2018_0815.pcap	7	5	71%

SAT-01-12-2018_0816.pcap	11	8	73%
SAT-01-12-2018_0817.pcap	33	31	94%
SAT-01-12-2018_0818.pcap	19	17	89%

A: #OF DETECTED PROTOCOLS IN TCP FLOW

B: #OF DETECTED PROTOCOLS IN REDUCED FLOW

C: DETECTION PERCENTAGE

Table 8 TCP-based Detection Applications and Reduction Rates

APPNAME	REDUCED	ORIGINAL	REDUCED	ORIGINAL	REDUCTION
Amazon	49.814	3.358.944	752	9.755	98,52%
CiscoVPN	240	360	4	6	33,33%
Cloudflare	3.432	57.108	52	290	93,99%
FTP_CONTROL	9.612	27.816	146	428	65,44%
Google	556.515	42.745.086	7.630	124.991	98,70%
HTTP	796.644	17.120.664	11.256	59.213	95,35%
HTTP_Proxy	240	360	4	6	33,33%
ICMP	532	1.024	6	12	48,05%
Microsoft365	264	20.034	4	40	98,68%
MsSQL-TDS	2.640	3.600	44	60	26,67%
Playstation	240	360	4	6	33,33%
RDP	480	600	8	10	20,00%
SMBv23	8.700	11.868	144	196	26,69%
SSH	253.900	7.116.484	3.404	46.886	96,43%
Telnet	6.600	8.040	110	134	17,91%
TLS	223.754	16.014.962	2.808	35.013	98,60%
UbuntuONE	1.510	3.991.232	20	3.188	99,96%

4.4. Experiment-3: Application Identification in full stream

In the final experiment, we treated the streams as is, sent them directly to the switch including all TCP and UDP traffic. Following results are achieved:

Table 9 Rates for Real-life Captures Using Full Streams

μ REDUCTION RATIO	84.73 %
μ REDUCTION FACTOR	6.5
μ DETECTION RATE	99.83 %

Table 10 Application Identification in Full Stream

APPNAME	REDUCED	ORIGINAL	REDUCED	ORIGINAL	REDUCTION
AFP	75.888	142.848	136	256	46,88%
Amazon	222.810	3.539.200	1.892	10.959	93,70%
AmongUs	74.772	187.488	134	336	60,12%
Ayiya	70.308	167.400	126	300	58,00%
BitTorrent	264.492	566.928	474	1.016	53,35%
BJNP	110.484	223.200	198	400	50,50%
CAPWAP	110.484	225.432	198	404	50,99%
CiscoVPN	90.636	174.456	166	318	48,05%
Cloudflare	3.432	57.108	52	290	93,99%
COAP	205.344	429.660	368	770	52,21%

Collectd	94.860	180.792	170	324	47,53%
CPHA	149.544	305.784	268	548	51,09%
DHCP	188.802	575.730	355	1.259	67,21%
DHCPV6	6.178	238.728	42	1.624	97,41%
DNS	1.285.798	1.528.164	11.150	12.354	15,86%
Dropbox	118.296	232.128	212	416	49,04%
EAQ	162.936	363.816	292	652	55,21%
Facebook	78.980	83.836	804	848	5,79%
FTP_CONTROL	9.736	27.940	148	430	65,15%
Github	8.592	8.986	92	96	4,38%
GMail	20.928	704.538	192	4.458	97,03%
Google	2.274.505	44.542.510	23.970	142.071	94,89%
GoogleServices	115.472	2.215.516	964	9.062	94,79%
GTP	263.376	565.812	472	1.014	53,45%
H323	159.588	351.540	286	630	54,60%
HTTP	799.416	17.123.436	11.300	59.257	95,33%
HTTP_Proxy	3.132	3.252	52	54	3,69%
IAX	118.296	241.056	212	432	50,93%
ICMP	380.064	6.251.658	4.052	48.536	93,92%
ICMPV6	5.548	88.904	62	954	93,76%
Instagram	74.948	77.950	484	512	3,85%
IPsec	279.632	590.996	500	1.058	52,68%
IRC	95.976	213.156	172	382	54,97%
iSCSI	212.040	449.748	380	806	52,85%
Kerberos	48.228	124.116	90	226	61,14%
LDAP	94.860	249.984	170	448	62,05%
LinkedIn	15.346	17.774	144	168	13,66%
LISP	156.240	330.336	280	592	52,70%
LLMNR	149.644	304.968	282	588	50,93%
MDNS	213.722	678.891	416	2.023	68,52%
Megaco	46.872	103.788	84	186	54,84%
Memcached	8.052	15.864	18	32	49,24%
Microsoft	76.694	784.104	640	2.493	90,22%
Microsoft365	5.064	144.776	44	314	96,50%
MsSQL-TDS	2.640	3.600	44	60	26,67%
NetBIOS	134.656	300.524	248	582	55,19%
NFS	111.600	243.288	200	436	54,13%
NTP	54.684	112.716	98	202	51,49%
OpenVPN	105.024	224.436	190	404	53,21%
OSPF	21.368	880.742	228	9.307	97,57%
Playstation	75.012	167.760	138	306	55,29%
Radius	213.156	444.168	382	796	52,01%
RDP	110.964	221.568	206	406	49,92%
Reddit	9.332	10.292	88	96	9,33%
RemoteScan	190.836	379.440	342	680	49,71%
RTSP	45.756	100.440	82	180	54,44%
RX	8.928.188	25.862.372	16.002	46.350	65,48%
sFlow	131.688	280.116	236	502	52,99%
SIP	245.320	512.044	446	924	52,09%
SMBv1	1.458	16.524	6	68	91,18%
SMBv23	9.192	12.360	152	204	25,63%
SOCKS	64.092	141.096	122	260	54,58%
SOMEIP	386.136	850.392	692	1.524	54,59%
SSDP	169.968	230.160	418	766	26,15%
SSH	254.024	7.116.608	3.406	46.888	96,43%
Starcraft	90.396	196.416	162	352	53,98%
Syslog	128.340	262.260	230	470	51,06%
TeamViewer	116.064	247.752	208	444	53,15%
Telnet	7.080	8.520	118	142	16,90%

Teredo	107.136	234.360	192	420	54,29%
TFTP	51.336	109.368	92	196	53,06%
TINC	100.440	223.200	180	400	55,00%
TLS	229.370	16.020.578	2.900	35.105	98,57%
Twitter	12.500	12.828	132	136	2,56%
UBNTAC2	106.020	233.244	190	418	54,55%
UbuntuONE	7.114	3.997.352	80	3.252	99,82%
VHUA	80.352	181.908	144	326	55,83%
Viber	686.340	1.487.628	1.230	2.666	53,86%
VMware	217.620	501.084	390	898	56,57%
Wikipedia	24.352	26.832	280	296	9,24%
WireGuard	112.716	255.564	202	458	55,90%
Xbox	229.896	510.012	412	914	54,92%
XDMCP	100.440	213.156	180	382	52,88%
YouTube	14.960	15.360	92	96	2,60%

4.5. Results and Discussion of the Experiments

The experimental study on the packet captures showed us that 2-packet reduction of a flow is possible to identify a flow.

The decrease in detection rate Table 4 is mostly caused by TLS encryption, which shows us that further study is needed to identify an encrypted flow. In addition to solve the problem coming through encrypted traffic, an ML-based approach would be implemented to succeed in the application identification of all flows. Based on the results from "**Table 5 Fully Detected Protocols from the capture files":**

- 125 out of 160 packet captures are correctly identified.
- 16 out of 160 packet captures could not be identified. Normally, 160 out of 160 packets would be identified correctly. 125 files identified correctly.
- 16 not identified at all (0 identification).
- 19 partially identified.
- 16 non-identified protocols are completely encrypted protocols.
- 125 identified protocols are mixed partially TLS and plain protocols.
- 19 partially identified protocols are mixed TLS and plain protocols partially.

Detection Rate drops with the reduced flow in encrypted traffic. (i.e., as we reduce the flow, we also lose important flow information that is needed for packet identification, short flows). The reason for not identifying these packet captures is they are mostly encrypted protocols, which require more than 2 packets to identify. We'll expand the experiments according to this.

In Experiment 2, the results in Table 6 showed that it is possible to increase the detection rate while the reduction rate is also increased. This is due to the fact that there are only 17 protocols detected in TCP streams, as indicated in Table 8 most of them are not TLS-based protocols or can be identified without deep inspection of the payload.

In Experiment 3, the results in Table 9 indicated that if we include UDP streams, the accuracy even goes higher, but the reduction rate decreases. This behavior is expected since the number of detected protocols in

Table 10 is 84, more than the number of applications detected in TCP streams, but the number of packets in UDP streams is lower than the number of packets in TCP streams. The decrease in reduction ratio is the result of the shorter flow size in UDP streams in the capture file.

CHAPTER 5

CONCLUSIONS

5.1. Main Conclusions

The main conclusion of this thesis is that application layer data processing can be performed with PISA switches at the network layer. We do not always need complex techniques to inspect the packets in L7; a simple flow-based packet reduction can achieve significant accuracy to identify the flows and add application-level visibility over the network. Data-stream processing combined with switch-level applications helps us building strong networking applications, such as DDoS attack detection mechanism. In-band Network Telemetry is in the central position of a programmable switch that distinguishes and separates them from the traditional switches. The proposed method constructs a Network Processor with a specific task from each PISA-stream processor pair. In other words, by using a single PISA switch and tens of stream processors with different features (DPI, NGFW, etc.) on different ports, our proposal constructs a big traffic exchange fabric with dynamically attached Network Processors of different types at a very low cost.

The result of this study demonstrates that the proposed system reduces the traffic load of such systems by a factor of 5.5 to 47 times with acceptable application identification. Applying ML-based approaches would increase the success rate of the proposed system with a margin of throughput needed compared to the legacy systems.

In addition, real traffic scenarios indicate that the performance gain would reach up to a factor of 40 on average by using the statistics in [86].

The studies in the literature and our experimental studies demonstrated that PISA switches are the glue for the SDN-NFV couple, increasing the performance of such systems. One of the major problems of the NFV systems is the performance bottleneck; however, the proposed solution also solves this problem for many use cases.

5.2. Future Studies

Encrypted network traffic identification with P4 language is one of the main future studies for this thesis, which is a very important topic for network security. In-band Telemetry seems to be a good place to start this study, as it tells us about the characteristics of a flow on a packet level. In this kind of analysis, AI/ML methods can provide great help in defining the features of traffic.

Another future area of interest could be Digital Twins (DT) in Telecommunication Networks. As PISA switches allow you to model the hardware in a software environment, it would straight-forward to build a DT of a telecom operator which needs to feed-forward the actual data and commands towards the active network.

REFERENCES

[1] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, “OpenFlow: Enabling Innovation in Campus NetworksSoftware-defined networking: A comprehensive survey,” *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, 2015, doi: 10.1109/JPROC.2014.2371999.

[2] ETSI, “Network Functions Virtualization,” 2020. [Online]. Available: https://www.etsi.org/technologies/nfv. [Accessed: 01-Dec-2020].

[3] K. Oztoprak, “fCDN:Geleceğin İletişim Dünyasında Enerji Verimli İçerik Dağıtım Sistemleri,” *J. Polytech.*, vol. 21, no. 4, pp. 999–1006, Oct. 2018, doi: 10.2339/politeknik.470675.

[4] K. Oztoprak, “mCSDN: A software defined network based content delivery system with D2D contribution,” in *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, 2016, pp. 2053–2057, doi: 10.1109/FSKD.2016.7603497.

[5] M. Pais, P. Debois, G. Kim, J. Humble, M. Hashimoto, and J. Allspaw, “Introducing DevOps to the traditional enterprise,” *InfoQ.com*, no. 14, p. 34, 2014.

[6] A. Henthorn-Iwane, “Why DevOps Is the Imperative Companion to SDN & NFV.” [Online]. Available: https://www.sdxcentral.com/articles/contributed/devops-sdn-nfv-imperative-companion-alex-henthorn-iwane/2015/09/. [Accessed: 13-Oct-2020].

[7] S. Conroy, “History of Virtualization,” 2018. [Online]. Available:

https://www.idkrtm.com/history-of-virtualization/. [Accessed: 13-Oct-2020].

[8] E. Brewer and J. Lin, “Application modernization and the decoupling of infrastructure services and teams,” 2019.

[9] R. Bauer, “What’s the Diff: VMs vs Containers,” *2018*. .

[10] K. Lane, “Kubernetes vs. Docker: What Does it Really Mean?,” 2020. [Online]. Available: https://www.sumologic.com/blog/kubernetes-vs-docker/. [Accessed: 13-Oct-2020].

[11] R. Morabito, J. Kjällman, and M. Komu, “Hypervisors vs. lightweight virtualization: A performance comparison,” *Proc. - 2015 IEEE Int. Conf. Cloud Eng. IC2E 2015*, no. March, pp. 386–393, 2015, doi: 10.1109/IC2E.2015.74.

[12] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, “An updated performance comparison of virtual machines and Linux containers,” in *2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2015, pp. 171–172, doi: 10.1109/ISPASS.2015.7095802.

[13] S. Shirinbab, L. Lundberg, and E. Casalicchio, “Performance evaluation of container and virtual machine running cassandra workload,” in *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*, 2017, pp. 1–8, doi: 10.1109/CloudTech.2017.8284700.

[14] Q. Zhang, L. Liu, C. Pu, Q. Dou, L. Wu, and W. Zhou, “A Comparative Study of Containers and Virtual Machines in Big Data Environment,” *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2018-July, no. July, pp. 178–185, 2018, doi: 10.1109/CLOUD.2018.00030.

[15] “eBrief: The Impact of Edge Computing on the Data Center,” 2019. [Online]. Available: https://www.sdxcentral.com/resources/sponsored/ebriefs/ebrief-

the-impact-of-edge-computing-on-the-data-center/. [Accessed: 14-Oct-2020].

[16] T. Zhang, “NFV Platform Design: A Survey,” pp. 1–29, 2020.

[17] H. Dnerta, “Vendor Independent SDN Architecture Solution,” pp. 1–6, 2020.

[18] K. M. Sadique, R. Rahmani, and P. Johannesson, “Identity Management in Internet of Things: A Software-Defined Networking Approach,” *Lect. Notes Electr. Eng.*, vol. 602, pp. 495–504, 2020, doi: 10.1007/978-981-15-0829-5_48.

[19] G. Antichi and G. Rétvári, “Full-stack SDN: The next big challenge?,” *SOSR 2020 - Proc. 2020 Symp. SDN Res.*, pp. 48–54, 2020, doi: 10.1145/3373360.3380834.

[20] S. Barguil, V. Lopez, and J. P. Fernandez-Palacios Gimenez, “Towards an Open Networking Architecture,” *2020 24th Int. Conf. Opt. Netw. Des. Model. ONDM 2020*, pp. 12–14, 2020, doi: 10.23919/ONDM48393.2020.9133038.

[21] ETSI, “GS NFV-MAN 001 Network Functions Virtualisation (NFV); Management and Orchestration,” Dec. 2014.

[22] L.F., “Open network automation platform,” 2020. [Online]. Available: https://www.onap.org/. [Accessed: 22-Oct-2020].

[23] ETSI, “Open source MANO (OSM) project,” 2020. [Online]. Available: https://osm.etsi.org/. [Accessed: 22-Oct-2020].

[24] Apache, “Open Baton: an open source Network Function Virtualisation Orchestrator (NFVO) fully compliant with the ETSI NFV MANO specification,” 2020. [Online]. Available: http://openbaton.org/. [Accessed: 22-Oct-2020].

[25] Cloudify, “Cloudify orchestration project portal,” *2020*. [Online]. Available: https://cloudify.co/. [Accessed: 22-Oct-2020].

[26] T. M. G.A. Carella, “Open baton: A framework for virtual network function management and orchestration for emerging software-based 5G networks.”

[27] G. M. Yilma, Z. F. Yousaf, V. Sciancalepore, and X. Costa-Perez, “Benchmarking open source NFV MANO systems: OSM and ONAP,” *Comput. Commun.*, vol. 161, no. July, pp. 86–98, 2020, doi: 10.1016/j.comcom.2020.07.013.

[28] A. Kapadia, “LF Networking: ‘Onboarding 5G CNFs with ONAP,’” *2020*. [Online]. Available: https://www.youtube.com/watch?v=SPiLpYjedIU&list=RDCMUCfX55Sx5hEFjoC3cNs6mCUQ&index=26&ab_channel=TheLinuxFoundation. [Accessed: 14-Oct-2020].

[29] B. Koley, “The Zero Touch Network,” 2016.

[30] ETSI, “GS ZSM 007 - V1.1.1 - Zero-touch network and Service Management (ZSM); Terminology for concepts in ZSM,” 2019.

[31] C. Benzaid and T. Taleb, “AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions,” *IEEE Netw.*, vol. 34, no. 2, pp. 186–194, 2020, doi: 10.1109/MNET.001.1900252.

[32] G. Carrozzo, “5G Enabling Technologies and Autonomic Networking,” *Wiley 5G Ref*, no. Mohr 2015, pp. 1–23, 2020, doi: 10.1002/9781119471509.w5gref133.

[33] GSMA, “AI in Network Use Cases in China,” no. October, 2019.

[34] ETSI, “GR ZSM 005 V1.1.1 Zero-touch network and Service Management (ZSM); Means of Automation,” 2020.

[35] ETSI, “GS ZSM 001 - V1.1.1 - Zero-touch network and Service Management (ZSM); Requirements based on documented scenarios,” 2019.

[36] M. Bunyakitanon, X. Vasilakos, R. Nejabati, and D. Simeonidou, "End-to-End Performance-Based Autonomous VNF Placement With Adopted Reinforcement Learning," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 2, pp. 534–547, 2020, doi: 10.1109/tccn.2020.2988486.

[37] A. Davis, J. Parikh, and W. E. Weihl, "EdgeComputing: Extending enterprise applications to the edge of the internet," *Proc. 13th Int. World Wide Web Conf. Altern. Track, Pap. Posters, WWW Alt. 2004*, pp. 180–187, 2004, doi: 10.1145/1013367.1013397.

[38] S. M. George, J. D. Ferguson, A. W. Weimer, and C. A. Wilson, "Patent Application Publication (10) Pub. No.: US 2004 / 0194691 A1," 2004.

[39] M. Theimer and M. B. Jones, "Overlook : Scalable Name Service on an Overlay Network Technical Report Microsoft Research Microsoft Corporation One Microsoft Way Overlook : Scalable Name Service on an Overlay Network," no. April, 2002.

[40] A. u. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A Survey of Mobile Cloud Computing Application Models," *IEEE Commun. Surv. Tutorials*, vol. 16, no. 1, pp. 393–413, 2014, doi: 10.1109/SURV.2013.062613.00160.

[41] L. B. Murali and N. Jayaveeran, "Cloud Computing – a buzz for IT Research," vol. XII, no. 0886, pp. 1327–1335, 2020.

[42] A. Reznik, "What is Edge?," 2018. [Online]. Available: https://www.etsi.org/newsroom/blogs/entry/what-is-edge. [Accessed: 22-Oct-2020].

[43] Q. Pham *et al.*, "A Survey of Multi-Access Edge Computing in 5G and Beyond: Fundamentals, Technology Integration, and State-of-the-Art," *IEEE*

Access, vol. 8, pp. 116974–117017, 2020, doi: 10.1109/ACCESS.2020.3001277.

[44] Y. Liu, M. Peng, G. Shou, Y. Chen, and S. Chen, “Toward Edge Intelligence: Multiaccess Edge Computing for 5G and Internet of Things,” *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6722–6747, Aug. 2020, doi: 10.1109/JIOT.2020.3004500.

[45] I. Morris, “ETSI Drops ‘Mobile’ From MEC,” 2016. [Online]. Available: https://www.lightreading.com/mobile/mec-(mobile-edge-computing)/etsi-drops-mobile-from-mec/d/d-id/726273. [Accessed: 29-Oct-2020].

[46] E. Banks, “Software-Defined WAN: A Primer,” 2014. [Online]. Available: https://www.networkcomputing.com/networking/software-defined-wan-primer. [Accessed: 22-Oct-2020].

[47] S. Troia, L. M. M. Zorello, A. J. Maralit, and G. Maier, “SD-WAN: An Open-Source Implementation for Enterprise Networking Services,” pp. 1–4, 2020, doi: 10.1109/icton51198.2020.9203058.

[48] J. Medved, R. Varga, A. Tkacik, and K. Gray, “OpenDaylight: Towards a model-driven SDN controller architecture,” *Proceeding IEEE Int. Symp. a World Wireless, Mob. Multimed. Networks 2014, WoWMoM 2014*, 2014, doi: 10.1109/WoWMoM.2014.6918985.

[49] L.F., “Open Daylight,” 2018. [Online]. Available: https://www.opendaylight.org/. [Accessed: 22-Oct-2020].

[50] X. Wu, K. Lu, and G. Zhu, “A survey on software-defined wide area networks,” *J. Commun.*, vol. 13, no. 5, pp. 253–258, 2018, doi: 10.12720/jcm.13.5.253-258.

[51] H. Attak *et al.*, “Guide to Security in SDN and NFV,” in *Computer*

Communications and Networks, 2017, pp. v–vii.

[52] A. Ali, R. Cziva, S. Jouët, and D. P. Pezaros, “{SDNFV}-Based {DDoS} Detection and Remediation in Multi-tenant, Virtualised Infrastructures,” in *Computer Communications and Networks*, Springer International Publishing, 2017, pp. 171–196.

[53] R. Tourani, A. Bos, S. Misra, and F. Esposito, “Towards security-as-a-service in multi-access edge,” *Proc. 4th ACM/IEEE Symp. Edge Comput. SEC 2019*, no. November, pp. 358–363, 2019, doi: 10.1145/3318216.3363335.

[54] NDN Consortium, “Named Data Networking,” 2020. [Online]. Available: https://named-data.net/. [Accessed: 22-Oct-2020].

[55] T. Combe, W. Mallouli, T. Cholez, G. Doyen, B. Mathieu, and E. Montes de Oca, “An SDN and NFV Use Case: NDN Implementation and Security Monitoring,” pp. 299–321, 2017, doi: 10.1007/978-3-319-64653-4_12.

[56] X. Wu *et al.*, “State of the art and research challenges in the security technologies of network function virtualization,” *IEEE Internet Comput.*, vol. 24, no. 1, pp. 25–35, 2020, doi: 10.1109/MIC.2019.2956712.

[57] P. Bosshart *et al.*, “Forwarding Metamorphosis: Fast Programmable Match-Action Processing in Hardware for SDN,” in *SIGCOMM Comput. Commun. Rev.*, 2013, vol. 43, no. 4, pp. 99–110, doi: 10.1145/2486001.2486011.

[58] C. Kim, “SLIDES: Programming The Network DataPlane: What, How, and Why?,” *Apnet*, 2017.

[59] S. Y. Wang, H. W. Hu, and Y. B. Lin, “Design and Implementation of TCP-Friendly Meters in P4 Switches,” *IEEE/ACM Trans. Netw.*, vol. 28, no. 4, pp. 1885–1898, 2020, doi: 10.1109/TNET.2020.3002074.

[60] Y. Yan, A. F. Beldachi, R. Nejabati, and D. Simeonidou, “P4-enabled Smart

NIC: Enabling Sliceable and Service-Driven Optical Data Centres," *J. Light. Technol.*, vol. 38, no. 9, pp. 2688–2694, 2020, doi: 10.1109/JLT.2020.2966517.

[61] C. Fernández, S. Giménez, E. Grasa, and S. Bunch, "A p4-enabled RINA interior router for software-defined data centers," *Computers*, vol. 9, no. 3, pp. 1–20, 2020, doi: 10.3390/computers9030070.

[62] R. Kundel *et al.*, "OpenBNG: Central office network functions on programmable data plane hardware," *Int. J. Netw. Manag.*, vol. n/a, no. n/a, p. e2134, doi: 10.1002/nem.2134.

[63] P. Bosshart *et al.*, "P4: Programming protocol-independent packet processors," *Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, 2014, doi: 10.1145/2656877.2656890.

[64] The P4 Language Consortium, "P4 16 Language Specification," 2019. [Online]. Available: https://p4.org/p4-spec/docs/P4-16-v1.2.0.html.

[65] "Getting started with P4 Language." [Online]. Available: https://p4.org/p4/getting-started-with-p4.html. [Accessed: 13-Feb-2021].

[66] Z. Hang, M. Wen, Y. Shi, and C. Zhang, "Programming protocol-independent packet processors high-level programming (P4HLP): Towards unified high-level programming for a commodity programmable switch," *Electron.*, vol. 8, no. 9, 2019, doi: 10.3390/electronics8090958.

[67] The P4.org Working Group, "In-band Network Telemetry (INT) Dataplane Specification," *The P4.org Applications Working Group*, 2020. [Online]. Available: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf. [Accessed: 22-Oct-2020].

[68] P. Parol, "P4 Network Programming Language—what is it all about?," 2020.

[Online]. Available: https://codilime.com/p4-network-programming-language-what-is-it-all-about/. [Accessed: 22-Oct-2020].

[69] A. Sgambelluri, F. Paolucci, A. Giorgetti, D. Scano, and F. Cugini, "Exploiting Telemetry in Multi-Layer Networks," pp. 1–4, 2020, doi: 10.1109/icton51198.2020.9203310.

[70] C. Kim *et al.*, "In-band Network Telemetry (INT) Information Processing INT Headers INT Header Types Handling INT Packets Header Format and Location INT over any encapsulation On-the-fly Header Creation Header Format Header Location and Format -- INT over Geneve Header Lo," 2016. [Online]. Available: https://p4.org/assets/INT-current-spec.pdf. [Accessed: 13-Feb-2021].

[71] A. Sari, A. Lekidis, and I. Butun, "Industrial Networks and IIoT: Now and Future Trends," in *Industrial IoT*, Springer, 2020, pp. 3–55.

[72] M. Zaharia *et al.*, "Apache Spark: A Unified Engine for Big Data Processing," *Commun. ACM*, vol. 59, no. 11, pp. 56–65, Oct. 2016, doi: 10.1145/2934664.

[73] Apache Foundation, "Apache Flink - Stateful Computations over Data Streams." [Online]. Available: https://flink.apache.org/. [Accessed: 13-Feb-2021].

[74] K. Oztoprak, "End-to-end visibility for an operator in the middle of Transformation," vol. XX, no. December, pp. 1–7, 2019.

[75] K. Oztoprak, "Subscriber Profiling for Connection Service Providers by Considering Individuals and Different Timeframes," *IEICE Trans. Commun.*, vol. E99.B, pp. 1353–1361, 2016, doi: 10.1587/transcom.2015EBP3467.

[76] K. Öztoprak, "Profiling subscribers according to their internet usage characteristics and behaviors," in *2015 IEEE International Conference on Big*

Data (Big Data), 2015, pp. 1492–1499, doi: 10.1109/BigData.2015.7363912.

[77] S. Narayana *et al.*, “Language-directed hardware design for network performance monitoring,” *SIGCOMM 2017 - Proc. 2017 Conf. ACM Spec. Interes. Gr. Data Commun.*, pp. 85–98, 2017, doi: 10.1145/3098822.3098829.

[78] A. Gupta, N. Feamster, R. Harrison, J. Rexford, M. Canini, and W. Willinger, “Sonata: Query-driven streaming network telemetry,” in *SIGCOMM 2018 - Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 357–371, doi: 10.1145/3230543.3230555.

[79] R. Teixeira, R. Harrison, A. Gupta, and J. Rexford, “PacketScope: Monitoring the packet lifecycle inside a switch,” *SOSR 2020 - Proc. 2020 Symp. SDN Res.*, pp. 76–82, 2020, doi: 10.1145/3373360.3380838.

[80] J. Hypolite, J. Sonchack, S. Hershkop, N. Dautenhahn, A. Dehon, and J. M. Smith, “DeepMatch: Practical deep packet inspection in the data plane using network processors,” *Conex. 2020 - Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.*, pp. 336–350, 2020, doi: 10.1145/3386367.3431290.

[81] Redis, “Redis open source (BSD licensed), in-memory data structure store, used as a database, cache, and message broker,” 2021. .

[82] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. Ghorbani, “Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy,” *2019 Int. Carnahan Conf. Secur. Technol.*, pp. 1–8, 2019.

[83] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, “nDPI: Open-source high-speed deep packet inspection,” in *2014 International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2014, pp. 617–622, doi: 10.1109/IWCMC.2014.6906427.

[84] M. A. Yazici and K. Oztoprak, “Policy broker-centric traffic classifier

architecture for deep packet inspection systems with route asymmetry," in *2017 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*, 2017, pp. 1–5, doi: 10.1109/BlackSeaCom.2017.8277681.

[85] K. Oztoprak and M. A. Yazici, "A hybrid asymmetric traffic classifier for deep packet inspection systems with route asymmetry," in *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, 2016, pp. 1–8, doi: 10.1109/PCCC.2016.7820616.

[86] P. Jurkiewicz, G. Rzym, and P. Boryło, "Flow length and size distributions in campus Internet traffic," *Comput. Commun.*, vol. 167, pp. 15–30, 2021, doi: 10.1016/j.comcom.2020.12.016.