

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA - UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS - CCT
MESTRADO EM COMPUTAÇÃO APLICADA**

NILTON JOSÉ MOCELIN JÚNIOR

**MULTI-DOMAIN QOS MANAGEMENT ARCHITECTURE FOR
SOFTWARE-DEFINED NETWORKS**

JOINVILLE

2025

NILTON JOSÉ MOCELIN JÚNIOR

**MULTI-DOMAIN QOS MANAGEMENT ARCHITECTURE FOR
SOFTWARE-DEFINED NETWORKS**

Dissertação submetida ao Programa de Pós-Graduação em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina, para a obtenção do grau de Mestre em Computação Aplicada.

Supervisor: Dr. Adriano Fiorese

JOINVILLE

2025

**Ficha catalográfica elaborada pelo programa de geração automática da
Biblioteca Universitária Udesc,
com os dados fornecidos pelo(a) autor(a)**

Mocelin, Nilton José Mocelin Júnior
MULTI-DOMAIN QOS MANAGEMENT ARCHITECTURE
FOR SOFTWARE-DEFINED NETWORKS / Nilton José
Mocelin Júnior Mocelin. -- 2025.
109 p.

Orientador: Adriano Fiorese Fiorese
Dissertação (mestrado) -- Universidade do Estado de
Santa Catarina, Centro de Ciências Tecnológicas, Programa
de Pós-Graduação em Computação Aplicada, Joinville, 2025.

1. Redes definidas por software. 2. Qualidade de serviço.
3. Classificação de tráfego. 4. Blockchain. 5. Gerenciamento
de rede multi-domínio. I. Fiorese, Adriano Fiorese. II.
Universidade do Estado de Santa Catarina, Centro de
Ciências Tecnológicas, Programa de Pós-Graduação em
Computação Aplicada. III. Título.

Nilton José Mocelin Júnior

Multi-domain QoS Management Architecture for Software-Defined Networks

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Computação Aplicada** área de concentração em "Sistemas de Computação", e aprovada em sua forma final pelo Curso de Mestrado em Computação Aplicada do Centro de Ciências Tecnológicas da Universidade do Estado de Santa Catarina.

Banca Examinadora:

PhD. Adriano Fiorese
Universidade do Estado de Santa Catarina (UDESC)

Membros:

Dr. Bruno Lopes Dalmazo
Universidade Regional de Blumenau (FURB)

Dr. Guilherme Piegas Koslovski
Universidade do Estado de Santa Catarina (UDESC)

Joinville, February 13th, 2025

Dedico este trabalho aos meus familiares, amigos, colegas e professores que me acompanharam e me deram forças nessa magnífica trajetória.

ACKNOWLEDGEMENTS

Gostaria de agradecer a todos os meus amigos, professores e colegas que estiveram comigo por toda essa jornada. Cada etapa foi uma conquista que não foi alcançada sem esforço e suporte.

“Independentemente das circunstâncias,
devemos ser sempre humildes, recatados
e despidos de orgulho.”

Dalai Lama

RESUMO

Embora a Internet conecte hosts ao redor do mundo, ela é uma vasta rede de redes gerenciada por vários Sistemas Autônomos (ASs), cada um com suas próprias políticas de roteamento e acordos de tráfego. Implementar qualidade de serviço (QoS) colaborativamente multi-domínios neste cenário é desafiador devido às complexidades da arquitetura de redes legadas e à falta de confiabilidade entre ASs. O surgimento da arquitetura de Redes Definidas por Software (SDN) trouxe novas possibilidades para o desenvolvimento de soluções de gerenciamento de redes. No entanto, existem questões em aberto quanto a implementação de QoS entre multi-domínios e a confiabilidade dos serviços prestados pelos provedores de rede. Ao analisarem os frameworks existentes na literatura, se percebe que ainda falta um sistema que englobe todas as fases da implementação de QoS multi-domínios, sendo elas: classificação de tráfego, reserva de recursos por fluxo, roteamento eficiente e auditoria de QoS. Neste trabalho, são propostas melhorias para o framework FLOWPRI-SDN 1.0, que implementa reserva de largura de banda priorizada multi-domínios. É proposta uma nova arquitetura que engloba a classificação do tráfego de dados em requisitos de QoS, juntamente com um mecanismo para colaboração multi-domínios, uma arquitetura de monitoramento de QoS e um mecanismo para auditar QoS fornecido, encapsulados como a versão FLOWPRI-SDN 2.0. Deste modo, foram conduzidos experimentos para avaliar diversos modelos de ML da literatura a fim de comparar o desempenho em cenários com diferentes quantidades de amostra. Resultados evidenciam que o tráfego de rede deve ser analisado em batch e não streaming. Como resultado, Random Forest alcança 98% de precisão na classificação de amostras de subfluxos de 10 pacotes em 8 classes de QoS. Em outro experimento, se demonstrou que isolar os fluxos por protocolo de transporte pode ser o melhor caminho, especialmente com o modelo XGBoost e para fluxos TCP, alcançando 99% de precisão em alguns casos. Outros componentes do Flowing with Priority (FLOWPRI)-SDN 2.0 foram avaliados em termos de sobrecarga de tempo, utilizando Linux NETNS como simulador. Por fim, foi mostrado que o tempo para se implementar QoS para um fluxo em um domínio leva cerca de 20ms e não é mais significativo que o tempo de atraso de enlace.

Palavras-chaves: Sistemas Autônomos, Reserva de Largura de Banda, Classificação de tráfego, Blockchain, Qualidade de Serviço e Gerenciamento Multi-domínio.

ABSTRACT

Although the Internet connects hosts around the world, it is a vast network of networks managed by several Autonomous Systems (ASs), each with its own routing policies and traffic agreements. Implementing quality of service (QoS) in a multi-domain collaborative manner in this scenario is challenging due to the complexities of the legacy network architecture and the lack of reliability between ASs. The emergence of Software Defined Networking (SDN) architecture has brought new possibilities for the development of network management solutions. However, there are open questions regarding the implementation of QoS across multi-domains and the reliability of the services provided by network providers. When analyzing the existing frameworks in the literature, it is clear that there is still a lack of a system that encompasses all phases of multi-domain QoS implementation, namely: traffic classification, resource reservation per flow, efficient routing and QoS auditing. In this work, improvements to the FLOWPRI-SDN 1.0 framework, which implements prioritized multi-domain bandwidth reservation, are proposed. A new architecture is proposed that encompasses the classification of data traffic into QoS requirements, together with a mechanism for multi-domain collaboration, a QoS monitoring architecture and a mechanism to audit provided QoS, encapsulated as the FLOWPRI-SDN 2.0 version. Thus, experiments were conducted to evaluate several ML models from the literature in order to compare their performance in scenarios with different sample sizes. Results show that network traffic should be analyzed in batch mode rather than streaming mode. As result, Random Forest achieves 98% accuracy in classifying subflow samples of 10 packets into 8 QoS classes. In another experiment, it was demonstrated that isolating flows by transport protocol may be the best approach, especially with the XGBoost model and for TCP flows, reaching 99% of precision in some cases. Other components of FLOWPRI-SDN 2.0 were evaluated in terms of time overhead using Linux NETNS as a simulator. Finally, it was shown that the time required to implement Quality of Service (QoS) for a flow within a domain is approximately 20ms and is not more significant than the link delay.

Key-words: Autonomous Systems, Bandwidth Reservation, Traffic Classification, Blockchain, Quality of Service and Inter-Domain QoS.

LIST OF FIGURES

Figure 1 – Internet Interconnections.	24
Figure 2 – IXP architecture.	26
Figure 3 – Google access over IPv6.	27
Figure 4 – ICMPv6 139 and 140: Node Information Query and Reply.	29
Figure 5 – SDN architecture layers.	29
Figure 6 – OpenFlow forwarding table.	32
Figure 7 – OpenFlow struct rules.	32
Figure 8 – Meter table.	33
Figure 9 – Meter bands.	34
Figure 10 – OVS architecture.	35
Figure 11 – QoS and Quality of Experience (QoE) in a network.	36
Figure 12 – Traffic monitoring.	38
Figure 13 – Blockchain architecture.	42
Figure 14 – The blocks of a blockchain structure.	43
Figure 15 – Hyperledger Sawtooth architecture.	44
Figure 16 – FLOWPRI-SDN.	45
Figure 17 – Link configuration.	46
Figure 18 – G-BAM flowchart.	47
Figure 19 – QoS rules.	48
Figure 20 – Switch management module.	48
Figure 21 – FLOWPRI-SDN 2.0 architecture.	60
Figure 22 – Switch management module.	61
Figure 23 – Switch management module.	62
Figure 24 – Creating Rules Process	66
Figure 25 – G-BAM flowchart.	68
Figure 26 – Packet-in process.	70
Figure 27 – Traffic classification architecture.	72
Figure 28 – Sawtooth Node Architecture	75
Figure 29 – Blockchain setup phase 1 and 2.	78
Figure 30 – Blockchain setup phases 3 and 4.	79
Figure 31 – Blockchain setup phase 5.	79
Figure 32 – Traffic Monitoring Architecture.	80
Figure 33 – Subflow filter.	83
Figure 34 – Traffic classification architecture.	87
Figure 35 – Network Topology for the experiments.	94
Figure 36 – A slice of the topology design, create with ip-netns.	95

Figure 37 – Flow classification diagram.	96
Figure 38 – Time from the first packet until QoS.	96
Figure 39 – Time for all domains to receive a Flow Requirements Description (FRED) and create the QoS rules.	97
Figure 40 – Time until the first QoS transaction.	98

LIST OF TABLES

Table 1 – QoS metrics for general network applications.	40
Table 2 – DSCP codes for real-time flows.	46
Table 3 – DSCP codes for non-real-time flows.	46
Table 4 – Comparison of related works of QoS frameworks.	58
Table 5 – Set of features calculated for classifying flows.	73
Table 6 – Application bandwidth and label defined.	74
Table 7 – Division of flows and packets in the experiment's database.	84
Table 8 – ML models comparison table (10pkts).	85
Table 9 – ML models comparison table (30pkts).	86
Table 10 – Flows distribution	88
Table 11 – Table of features part 1.	89
Table 12 – Table of features part 2.	90
Table 13 – Results for Phase 1	92
Table 14 – Results for Phase 2	93

LIST OF ABBREVIATIONS AND ACRONYMS

API	Application Programming Interface
AS	Autonomous System
ASN	Autonomous System Numbe
BAM	Bandwidth Allocation Method
BGP	Border Gateway Protocol
DPI	Deep Packet Inspection
DSCP	Differentiated Services Code Point
E2E	End-to-End
FLOWPRI	Flowing with Priority
FRED	Flow Requirements Description
G-BAM	Generalized Bandwidth Allocation Method
HTB	Hierarchical Token Bucket
HTTP	Hypertext Transfer Protocol
IANA	Internet Assigned Numbers Authority
IAT	Inter-arrival Time
ICMP	Internet Control Message Protocol
ID	Identification
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IRR	Internet Routing Registry
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
IXP	Internet Exchange Point

JSON	Java Script Object Notation
KPI	Key Performance Indicators
LACNIC	Latin American and Caribbean Internet Addresses Registry
MOA	Massive Online Analysis
ML	Machine Learning
MPLS	Multiprotocol Label Switching
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NI	Node Information
OSI	Open System Interconnection
OSPF	Open Shortest Path First
OVS	Open Virtual Switch
OVSDB	Open Virtual Switch Database
PBFT	Practical Byzantine Fault Tolerance
PoET	Proof of Elapsed Time
PoW	Proof of Work
QoE	Quality of Experience
QoS	Quality of Service
RIP	Routing Information Protocol
RIR	Regional Internet Registry
SDN	Software-Defined Network
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VoIP	Voice over Internet Protocol (IP)
VPN	Virtual Private Network

CONTENTS

1	INTRODUCTION	16
1.1	OBJECTIVES	20
1.1.1	GENERAL OBJECTIVE	20
1.1.2	SPECIFIC	20
1.1.3	CONTRIBUTIONS	21
1.2	WORK STRUCTURE	21
2	BACKGROUND	22
2.1	The Internet	22
2.1.1	Autonomous Systems Interconnections	24
2.2	IPv4 vs IPv6	27
2.2.1	ICMPv6	28
2.3	Software-Defined Networking	29
2.3.1	Controller	30
2.3.2	Open Flow	31
2.3.2.1	OpenFlow Tables	31
2.3.2.2	Meter Table	33
2.3.3	Open vSwitch	34
2.4	Quality of Service	36
2.4.1	Traffic Monitoring	37
2.4.2	Applications over the Network	38
2.5	Identifying the traffic flow requirements	39
2.6	Blockchain and QoS Trustness	41
2.6.1	Hyperledger Sawtooth Framework	43
2.7	FLOWPRI-SDN Framework	44
2.7.1	Generalized Bandwidth Allocation Method	47
3	RELATED WORK	50
3.1	QoS in SDN	50
3.2	Traffic Classification in SDN	54
3.3	Blockchain and QoS	55
3.4	Considerations About The Related Work	56
4	PROPOSAL	59
4.1	FLOWPRI-SDN 2.0	59

4.1.1	Switches Management	60
4.1.2	FRED	62
4.1.2.1	Multi-domain communication: Sending FREDS	63
4.1.3	Local Resource Management	64
4.1.3.1	QoS Monitoring Rules	65
4.1.3.2	G-BAM	67
4.1.3.3	Backbone G-BAM	68
4.1.4	Packet-in event handler	69
4.1.4.1	Handling unmatched packets	70
4.1.4.2	Handling ICMPs	71
4.2	Traffic Classification	71
4.3	QoSBlockchain	73
4.3.1	QoSBlockchain architecture	74
4.3.2	QoSBlockchain Setup	77
4.4	Network Monitoring	79
4.5	Requirements and Restrictions	81
4.6	Chapter Considerations	81
5	EXPERIMENTS	83
5.1	Streaming data vs Batch data	83
5.2	Refining the Traffic Classification Architecture	86
5.2.1	Traffic collection and Processing phases	88
5.2.2	Training Phase	91
5.2.3	Results	91
5.3	FLOWPRI-SDN 2.0 Overhead	93
5.3.1	Building the Topology	94
5.3.2	Flow Classification Overhead	95
5.4	Chapter Considerations	98
6	FINAL CONCLUSIONS	100
	BIBLIOGRAPHY	102
	APPENDIX A – FLOWPRI-SDN 2.0 SOURCE CODE	108

1 INTRODUCTION

The Internet is a vast network that interconnects other networks and enables end-devices to communicate over long distances. Initially, the Internet was designed to provide best-effort service for data flows. As it was expanding, numerous independent administrative domains came to own the networks that compose the Internet (GAO, 2001). Subsequently, protocols and tools emerged to allow differentiated treatment of flows and enhance communication quality, such as DiffServ and IntServ (BABIAZ K. CHAN, 2006; BRADEN et al., 2000). However, to this day, there are no globally accepted protocols to implement multi-domain QoS and ensure that networks meet the minimum requirements of end-to-end flows.

In general terms, QoS can be defined as the set of characteristics of a telecommunication service that ensure the ability to meet the announced and implied requirements of the service users (JANEVSKI; JANKOVIC; MARKUS, 2017). Providing network resource management, especially bandwidth management, brings several benefits to network administrators. It ensures that critical services are not affected during network congestion. Additionally, it helps network administrators to allocate available network resources effectively and to avoid unnecessary infrastructure expenses.

Even with few solutions for QoS, the evolution of the Internet has contributed to the emergence of new services such as Voice over IP (VoIP), video streaming, and online gaming (AL-HARBI et al., 2021). Inevitably, the applications of these types, are designed to operate under specific network conditions. Some are highly sensitive to QoS metrics (bandwidth, delay, jitter, and loss), especially real-time and multimedia applications and others do not. So, they can be treated as best-effort traffic. However, these application usually compete for bandwidth across the network without guarantees for their requirements in a best-effort way.

This diversity of applications and requirements has presented new challenges to network providers. To provide per-flow QoS to such applications spanning multiple administrative domains demands fair and robust coordination (PODILI; KATAOKA, 2021). In the Internet, there are no default protocol to identify applications and requirements in real-time. As mentioned before, DiffServ and IntServ were proposed to help to implement different QoS actions for traffic flows by reading specific codes in the Differentiated Services Code Point (DSCP) field of Internet Protocol Version 4 (IPv4) packets. However, it is not explored how to classify flows into codes that represent their resource requirements or into applications (JANEVSKI; JANKOVIC; MARKUS, 2017).

Identifying the traffic is the first step to provide QoS for a flow. Therefore, to

implement the desired traffic classification, there are basically four ways: self identification, packet inspection, port based and statistical analysis. All of these methods have pros and cons. However, the simplicity and efficiency of Machine Learning (ML) models has proved they are a good option to explore (SERAG et al., 2024; AMARAL et al., 2016). ML has already been explored for traffic classification, but not necessarily they are integrated into network management test scenarios.

Another aspect of QoS provisioning is QoS traceability. In the last years, the amount of Internet Service Provider (ISP) has increased, but the services offered did not, and it is not easy for customers to identify the QoS provided for their traffic flows. Also, ISPs and Autonomous System (AS)s are autonomous, so each one have their own management policy. Therefore, the Internet lacks of trust in the relationship of customers and administrative domains, and transparency in the services that are being paid (PODILI; KATAOKA, 2021).

For this reason, trust between administrative domains is important (PODILI; KATAOKA, 2021). Trust awareness reduces the risk of violating QoS and enhances confidence in operating QoS across multiple domains. Similarly, ensuring auditability of the results of QoS implementation within each domain is crucial to verify QoS compliance and subsequently assess domain trustworthiness by demonstrating that QoS was indeed delivered as agreed upon by the administrative domains of a flow's route. This capability can position QoS as a viable paid service. These factors can drive the adoption of new mechanisms for cross-domain QoS, thereby improving overall network quality and creating new revenue streams for ASs.

The issue of inter-domain trust for End-to-End (E2E) QoS provisioning is still very unexplored field. Recently, blockchain has emerged as one alternative for mitigating trust issues. As blockchains are consistent and distributed databases, traffic can be monitored to analyze current traffic flow quality, so this information can be stored (KARAKUS; GULER; ULUDAG, 2021). Many works found in the literature are limited to store routing information like link utilization (KARAKUS; GULER; ULUDAG, 2021; PODILI; KATAOKA, 2021; RAHMAN et al., 2021). However, there is a lack of works dealing with trust between network domains, particularly storing historical resource allocation information for application flows between the domains.

As these new challenges emerge, the old legacy network paradigm becomes inefficient to handle the required dynamism. However, the Software-Defined Network (SDN) architecture brought the concept of programmable networks by separating the control plane from the data plane ((ONF), 2012). In this paradigm, forwarding devices are dynamically programmed to take forwarding actions on the packet flows they process. The controller is the device that manages network control and makes decisions about actions to be taken by each forwarding device under its management. The con-

troller communicates with forwarding devices using Open Flow communication protocol, which defines a series of events and messages for this purpose. This approach enables the implementation of much more powerful and dynamic network management strategies compared to the conventional paradigm.

One existing controller framework is the Flowing with priority in SDN (FLOWPRI-SDN) (JÚNIOR; FIORESE, 2023). It allows per-flow QoS provisioning between multiple administrative domains (AS, ISPs or other entities). For doing so, it uses a QoS contract, which is informed by hosts, about the requirements of their flows. The FLOWPRI-SDN distributes the QoS contract to other controllers on the route the flow is sent. Along with the QoS identification, the FLOWPRI-SDN provides a bandwidth allocation strategy, based on Generalized Bandwidth Allocation Method (G-BAM).

In the FLOWPRI-SDN was proposed the G-BAM, that rules where to allocate a traffic flow into queues reserving bandwidth. In this module, all links of the switches have the bandwidth split into four classes using the Hierarchical Token Bucket (HTB) queues: control, best-effort, real-time, non-real-time. This strategy allows to separate traffic with bandwidth reservation from best-effort and also, allows the best-effort class to borrow bandwidth from QoS classes. The HTB queues only share bandwidth that is not being used. Whereas, with the FLOWPRI-SDN and Open Flow meter rules, the QoS flows have bandwidth reserved in their respective queue, since the meter rules shape the traffic and the controller secures that only the available amount of bandwidth is allocated. Then, there is no dispute over resources between the QoS flows.

A network simulator was implemented using Linux NETNS (KERRISK, 2013). It was designed a topology using virtual namespaces and virtual interfaces. Then, the FLOWPRI-SDN 2.0 was evaluated managing this topology, in Chapter 5.

The general context of multi-domain QoS implementation and the presented gaps lead to the following research questions:

R1: Which ML algorithms are most suitable for traffic classification based on QoS statistics of traffic flows?

R2: Which amount of packets are enough for performing traffic classification ?

R3: How to integrate traffic classification in the FLOWPRI-SDN architecture.

R4: How to perform traffic monitoring and QoS accountability in SDN ?

R5: What is the overhead of the modules implemented in the FLOWPRI-SDN 2.0 ?

Thus, this study proposes to implement QoS as a service, addressing dynamic resource provisioning, the needs for multi-domain ISPs trustness, and, handling the application traffic classification and monitoring. Together, all these components form

the FLOWPRI-SDN framework version 2.0. The main features of the new framework are as follows:

- *Intra-Domain Resource Management.* In FLOWPRI-SDN 2.0, bandwidth is the most important resource, along with traffic priority. This way, each switch port has bandwidth of their links separated in classes by using hierarchical queues. Every time one data flow sends packets, and they arrive in the controller, the FLOWPRI-SDN 2.0 chooses the right route for them and allocates bandwidth depending on their QoS, along with the priority, if the devices of the route can provide these resources. The FLOWPRI-SDN 2.0 also manages the available bandwidth between the QoS classes of a link, sharing bandwidth between them according to the flow priority or denying the QoS and sending traffic by best-effort. In the case of QoS avoidance, then after a couple of seconds, that flow can try to get QoS again.
- *Multi-domain QoS allocation.* The edge networks can flood the core network with data flows of different QoS requirements. FLOWPRI-SDN 2.0 mitigates the degradation of the QoS by sharing the QoS requirements of the flows towards the domains of the route they will follow. This way, each FLOWPRI-SDN 2.0 domain must provide the resources needed for the data flows along with monitoring the QoS provided and communicate the QoS state in case of degradation.
- *Autonomous Systems are still autonomous.* FLOWPRI-SDN doesn't influence on the autonomy of the domains, because it only manages resources. As providing the right amount of resources per flow can be benefit for both, network domains and clients, it becomes a motivation to implement the framework. Also, each Autonomous System can still decide their routing policies, that is what makes them autonomous.
- *Flow Priority.* In FLOWPRI-SDN, each class of QoS supports three priority levels. Then, according to the QoS requested, the priority and the kind of application, one data flow can have priority in the bandwidth allocation process, even removing less priority flows to allocate the most priority. It is up to the administrator domain to select the priority of the data flows otherwise all of them are low priority by default.
- *Network Traffic Identification.* The proposed framework implements machine learning to classify traffic flows into service classes and application classes. This way, they are mapped into quality levels and QoS requirements, such as: bandwidth, delay, jitter and loss.

- *Network Traffic Monitoring.* Network monitoring is a required functionality to implement these components, because traffic classification and QoS blockchain depend on flow analysis. The monitored traffic statistics of QoS-sensitive flows can be shared between edge FLOWPRI-SDN domains and end-hosts and stored in distributed blockchains.
- *QoS auditability and trustability.* In the classical Internet architecture, hosts are not aware of the QoS provisioning. With FLOWPRI-SDN, there is a service for the hosts to monitor the real QoS provided. In the same way, the edge frameworks in the route of a flow monitors the traffic QoS provided. Then, these records are stored in a distributed blockchain database that helps to keep track of the QoS provisioning and improves trustability between the network entities and hosts.

1.1 OBJECTIVES

This section presents the general and specific objectives of this work.

1.1.1 GENERAL OBJECTIVE

With the FLOWPRI-SDN 2.0, the main goal is to improve and develop methods for performing multi-domain resource reservation based on flow QoS requirements. This includes enhancing motivation and trustworthiness for QoS provisioning between network domains and clients. Additionally, this work aims to evaluate the impact of the proposed methodologies in terms of overhead and scalability, and to compare FLOWPRI-SDN mechanisms with other frameworks.

1.1.2 SPECIFIC

The specific goals are outlined as follows:

1. To implement an architecture for efficient traffic classification.
2. To Seek improvements for the overall traffic classification process, such as exploring new ways to improve the overall traffic classification process.
3. To implement a mechanism for QoS control audition using blockchain.
4. To enhance communication mechanisms between host-controller and controller-controller.
5. To investigate the impacts in terms of network and processing overhead of FLOWPRI-SDN components.
6. To study the positive impacts of QoS provisioning mechanisms.

7. To explore suitable implementation scenarios for FLOWPRI-SDN and identify its limitations.
8. To investigate the implementation of auditability via distributed blockchain database for provisioned QoS on application data flows.

1.1.3 CONTRIBUTIONS

In this work, a traffic classification architecture is presented to identify flow types and application requirements. Additionally, an inter-domain communication protocol is proposed to facilitate the announcement and monitoring of QoS requirements. Finally, QoS monitoring statistics are stored in a distributed blockchain database, called QoSBlockchain, enhancing trust in collaborative QoS provisioning efforts. This work also realizes a proof of concept on utilizing blockchain as a distributed database for bandwidth reservation.

Moreover, some papers were published upon this work's writing, cited as follows: (JR; PARPINELLI; FIORESE, 2023), (JÚNIOR; FIORESE, 2023), (JR.; PARPINELLI; FIORESE, 2023), and an upcoming paper to be released on June, 25th in Hybrid Intelligent Systems, entitle as "A Comparison Between Traffic Classification Models for Bandwidth Management in Software-Defined Networks".

1.2 WORK STRUCTURE

This work is structured as follows: Chapter 2 provides the necessary theoretical background to support understanding of the design decisions made in the development of FLOWPRI-SDN 2.0 and its proposed components. It includes an overview of the current state of the Internet and the autonomous entities that comprise it, an exploration of SDN architecture, and definitions and techniques for QoS provisioning through resource reservation. Chapter 3 reviews previous related work, discussing their approaches to QoS provisioning. Chapter 4 introduces the FLOWPRI-SDN 2.0 framework and the proposed features. Chapter 5 details tests and experiments related to FLOWPRI-SDN 2.0. Finally, Chapter 6 presents the last considerations for this study.

2 BACKGROUND

This chapter contains the theoretical foundation necessary to understand the elements that compose the solutions developed in the FLOWPRI-SDN framework proposal. First, this work presents the current state of the Internet, as well as the relationship between ASs and their traffic exchange agreements. Next, it introduces the SDN network architecture and the key components used to develop QoS provisioning solutions. Following that, it discusses the differences between Internet Protocol Version 6 (IPv6) and IPv4, highlighting the IPv6 control protocol, which allows for information exchange between devices in the network. Afterward, it explores key aspects related to QoS provisioning, including traffic classification in the network and how consensus mechanisms can be used in the context of QoS.

2.1 THE INTERNET

The Internet is a very big network of interconnected computer networks that started largely as a public sector endeavor, but subsequently became increasingly commercialized (BADASYAN; CHAKRABARTI et al., 2003). Nowadays, the Internet connects thousands of ASs operated by many administrative domains such as ISPs, companies and universities (GAO, 2001; JANEVSKI; JANKOVIC; MARKUS, 2017).

The ISPs are companies or organizations that offer various Internet-related services to their subscribers. They can be privately owned companies, government-owned entities, or a combination of both. The primary goal of an ISP is to establish and maintain a network infrastructure that allows users to connect to the Internet, exchange data, and access online services. Apart from that, they can also provide services like e-mail hosting, website hosting, domain registration, Virtual Private Network (VPN)s, and more (ALLEN, 2023; SHOKOUHYAR et al., 2021; JANEVSKI; JANKOVIC; MARKUS, 2017).

The network of ISPs can be divided into core network and access network. The ISPs' core network forms the backbone infrastructure, and it is responsible for routing and transmitting data across long distances. The backbone's hardware is composed of high-capacity routers, fibre-optic cables and high bandwidth to enable efficient transport of data packets between different geographical locations. The access networks, on the other hand, connects end-users, like business or individuals, to the ISP's core network. They use similar hardware but with limited bandwidth to allow users to access online resources and services through the core networks (ALLEN, 2023; METZ, 2001; GREENSTEIN, 2020; JANEVSKI; JANKOVIC; MARKUS, 2017).

An AS on the Internet is a group of networks administered by one or more network operators that are ruled by the same routing policy. They serve as the building blocks of the Internet, facilitating the seamless flow of data packets between different networks (LACNIC, 2023). In the Latin America region, Latin American and Caribbean Internet Addresses Registry (LACNIC) is the Internet Routing Registry (IRR) in charge of allocating IPv4 and IPv6 network blocks, and also Autonomous System Number (ASN) to the administration domain networks. The ASN was first defined at RFC1930 and improved at RFC4893 to be an 16-bit or a 32-bit identifier number and must only be assigned when a new routing policy is necessary. It is possible for groups of networks that are not under the same management to share the same ASN, but it will require an additional coordination among the network administrators (LACNIC, 2023). Then, to an organization to get an ASN and become AS, it must meet some requirements, such as:

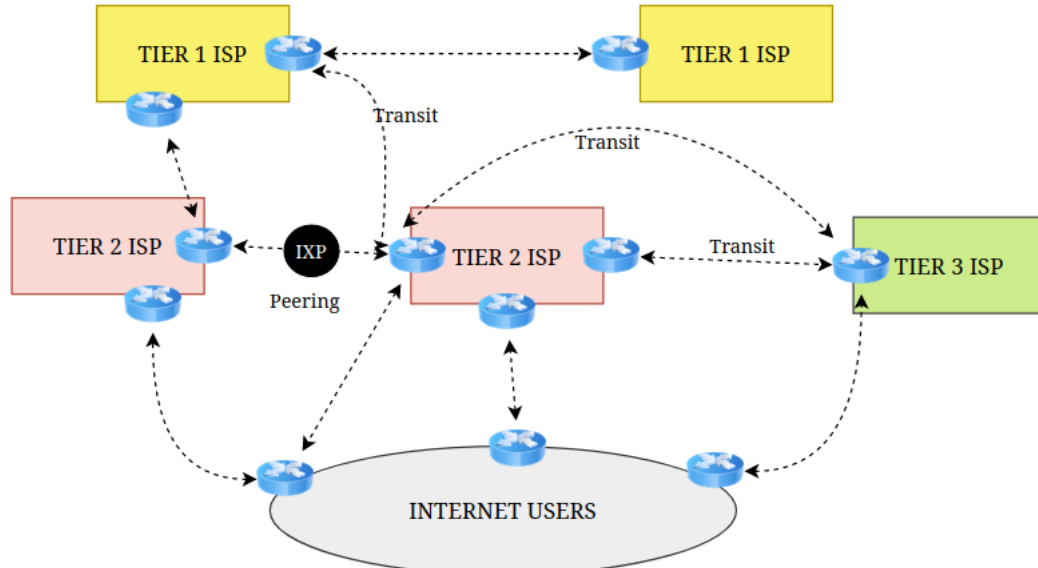
- The organization must have the need to interconnect with other independent ASs. If the number is not being used for a period longer than six months, LACNIC may revoke the assigned ASN.
- It must have Border Gateway Protocol (BGP) compatible hardware and software.
- It must be multi-homed: At least two connections with the Internet or one Internet connection along with one Internet Exchange Point (IXP) connection.
- It must have at least a /20 network block, and have 25% of the IP addresses allocated.
- It must detail the routing policy, specifying the ASNs with which the organization will interconnect and the IP address that will be announced.
- It must maintain contact with LACNIC and updated policy records.

There are some fundamental differences between ISPs and ASs. ISPs can be ASs but not all ASs are ISPs. In fact, one ISP can operate one or several ASs. This offers ISPs greater control and flexibility in managing their network traffic, optimizing the delivery of data packets based on various factors such as network congestion, distance, and reliability (CACHEFLY, 2023).

Autonomous Systems (ASs) are ranked based on their cone numbers. The cone size of an AS represents its size in terms of the number of reachable IP addresses within the AS, as well as the number of other ASs that can be reached through it. It takes into account the network infrastructure, size, and level of connectivity. Based on the cone size and the role of the AS, these networks can be categorized into three tiers

(ALLEN, 2023; HENTHORN-IWANE, 2024). Figure 1 shows how the administrative domains interconnects users to form the Internet:

Figure 1 – Internet Interconnections.



Source: Adapted from (PHOKEER, 2013).

- Tier 1 Networks: The largest ASs that form the backbone of the Internet. They have extensive networks and significant network capacity and high-speed connections, allowing them to handle large volumes of traffic. Tier 1 networks can also peer with each other without paying for traffic exchange.
- Tier 2 Networks: The regional or national providers that connect to Tier 1 ISPs for Internet transit. They actually have substantial network infrastructure but rely on Tier 1 providers for broader connectivity.
- Tier 3 ISPs are local or niche providers that offer Internet services within limited geographic areas or for specific market segments. They often rely on upstream providers, such as Tier 1 or Tier 2 ISPs, for Internet connectivity. Tier 3 ISPs may focus on specialized services, such as community networks or niche markets. These ISPs may play an essential role in connecting remote or underserved areas.

2.1.1 Autonomous Systems Interconnections

Routing within an AS is managed by intra-domain routing protocols such as static routing, Open Shortest Path First (OSPF), Intermediate System to Intermediate System (IS-IS), and Routing Information Protocol (RIP). A pair of ASs interconnects

via dedicated links and/or public network access points. Routing between ASs is governed by the inter-domain routing protocol BGP. A crucial characteristic of BGP is its allowance for each AS to set its own administrative policies in route selection, route announcement, and route acceptance. Commercial contractual relationships between administrative domains play a significant role in shaping these routing policies (GAO, 2001).

Most of the routing protocols, such as those cited before, are traffic insensitive. They will route the traffic along the most congested link on the Internet because they do not route around congestion. In case of congestion, a legacy domain cannot be able to provide QoS for the traffic. Moreover, neighbors and hosts will never know about it as well (COMPUTERPHILE, 2016).

The information exchange by the BGP protocol depends on the commercial relationships between ASs (GAO, 2001). These settlements are not regulated by a central authority, but are implemented on a case-by-case basis by the operators, and generally, minding the economical aspect. These commercial agreements can be classified into customers-provider, peering, mutual-transit, and mutual-backup agreements.

A customer pays its provider for connectivity to the rest of the Internet, representing the customer-provider relationship. Therefore, a provider does traffic transit for its customers (GAO, 2001). Another model is when a pair of peers agree to exchange traffic between their respective customers free of charge (peering).

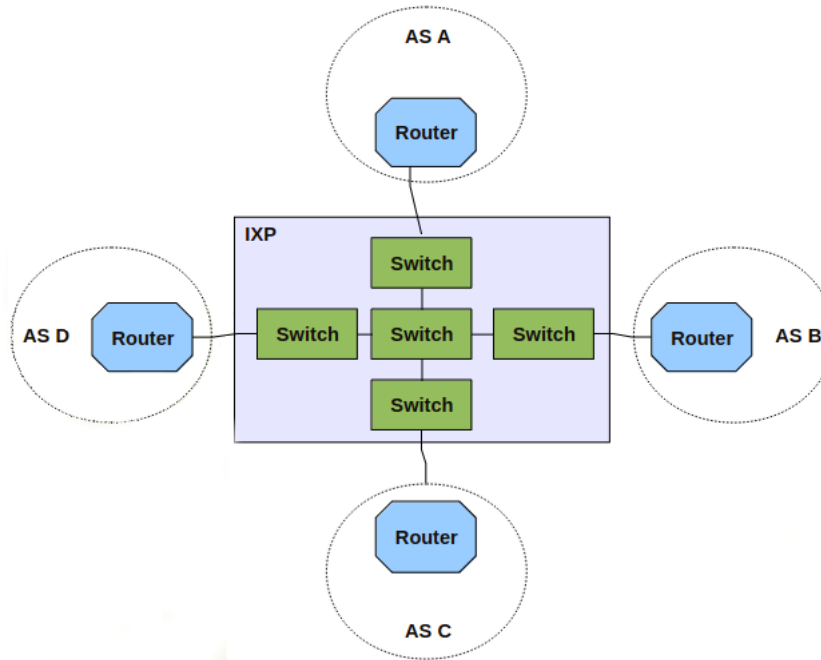
A mutual-transit agreement allows a pair of administrative domains to provide connectivity to the rest of the Internet for each other. This mutual-transit relationship is typically between two administrative domains such as small ISPs who are located close to each other and who cannot afford additional Internet services for better connectivity. A pair of administrative domains may also provide backup connectivity to the Internet for each other in the event that one administrative domain's connection to its provider fails (mutual-backup). These contractual commercial agreements between administrative domains play a crucial role in shaping the structure of the Internet and the end-to-end performance characteristics (GAO, 2001).

Autonomous Systems can also interconnect at dedicated point-to-point links or public IXPs. Public exchange points typically consist of a shared medium that interconnects routers from several ASs, physically. However, the connectivity at the IXP does not necessarily imply that every pair of ASs exchanges traffic with each other (GAO, 2001).

In Brazil, the Internet Management Committee promotes the creation of IXPs. Figure 2 depicts this solution and shows the objective of enabling direct connection between the entities that make up the Internet (REIS, 2010). An IXP optimizes inter-

connections between ASs, as it enables: Better quality (lower latency), avoids external intermediaries, lower costs and improves the organization of the regional network structure (concentration points).

Figure 2 – IXP architecture.



Source: (REIS, 2010)

It can be noticed that most of the inter-AS agreements are motivated by profit. ASs will prefer to exchange traffic with partners in a longer route than with the domains from a shorter one, if it is cheaper. This approach ignores the requirements of the application data traffic that cross the network. However, if at some point QoS for the applications become a service, the ISPs and ASs will be more interested to work together to provide QoS end-to-end.

The best-effort transit model systems leaves the networks vulnerable to malicious exploits and denial-of-service attacks. In the “best-effort” model, a computational resource does its best to satisfy service requests from users as long as the resource is not fully utilized. That is, the resource is available to use regardless of the state of resources, leading to a danger of completely depleted resources by malicious attackers, for example, a denial-of-service attack (YE, 2002).

One other problem of not implementing QoS policies, is the resource over-provisioning. The ISPs billing model are basically of three kinds: flat rate, bandwidth-based and data-based. In the flat rate mode, customers pay a fix amount every month for an explicitly upper bandwidth bound. The bandwidth-based defines that a customer is charged for a certain amount of bandwidth and pays extra for the bandwidth exceeded. The data-based is similar to the bandwidth-based approach, except that cus-

tomers are charged per megabytes and not for megabit per second (XIAO et al., 2002). All of these approaches doesn't take the QoS of the applications in account, and can induce to over provisioning resources.

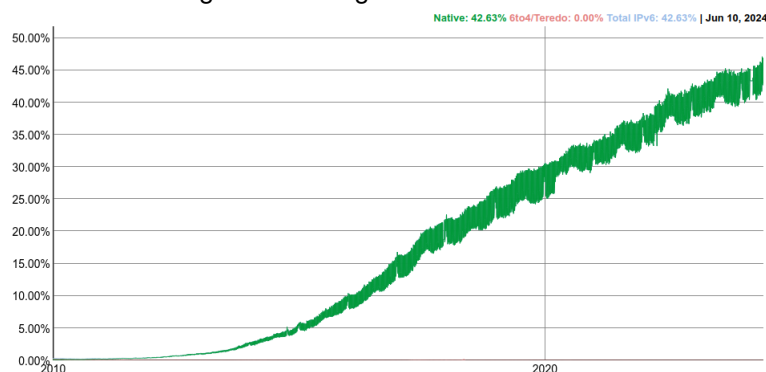
2.2 IPV4 VS IPV6

When the Internet was first proposed, QoS and massive scalability were secondary aspects. The IP is a network layer protocol that keeps the Internet working together. This protocol was developed to provide interconnection between hosts and networks, offering a best-effort transport method to transport datagrams from source to destination. IPv4 is the most common version yet utilized over the networks, but it is supposed to be gradually replaced by the IPv6 version along the next years (DAS et al., 2023).

An IPv4 address representation has 4 octets of 8 bits separated by a dot, making it 32 bits long. This composition supports 2^{32} or 4,294,967,296 unique addresses that are distributed by the Regional Internet Registry (RIR) of the region. Actually, the IPv4 addresses are globally sold out. In Latin America region, only organizations that have never requested IPv4 blocks can get it, and they are limited to /22 blocks, or 1024 unique addresses (NIC.BR, 2022).

The IPv6, on the other hand, has a way larger range of unique addresses. The address structure is composed of 8 groups of 16 bits separated by a colon, that totals 128 bits in length. This allows 2^{128} or 340 trillion of unique addresses, this will be enough for Internet addressing for a long time. Figure 3 shows that IPv6 is getting more attention recently, where 42.63% of the users that access Google services nowadays are using IPv6.

Figure 3 – Google access over IPv6.



Source: (GOOGLE, 2024).

The vast number of unique IP addresses provided by IPv6 reduces the necessity of using Network Address Translation (NAT). NAT is a mechanism that translates one IP address into another, commonly used to extend the limited number of IPv4 ad-

addresses within a network domain. This allows a domain to translate a public IP address obtained from a RIR into a private address visible only within the domain. However, NAT has weaknesses such as the complexity of modifying packet headers and the fact that the IPv4 address of a packet sent by a host in one network domain will have different source and destination IP addresses in another network domain. This complexity can make the implementation of QoS per flow more challenging.

IPv6 has more support to QoS implementation, it renames the DSCP header field to traffic class (8 bits) and adds a new one, called flow label (20 bits). Another difference between IPv4 and IPv6 is that, IPv4 routers can make packet fragmentation. It means that if one packet is bigger than the domain's Maximum Transmission Unit (MTU), it can be separated in two (or more) packets by some router and keep being sent to the destination. Whereas, in IPv6 version, the fragmentation must be done by the sender. This implies that sometime the networks will have to support jumbo frames (changing the default MTU from 1500 bytes to 9000 bytes) to avoid asking the sender to adjust their parameters and send everything again (DAS et al., 2023; SANTOS, 2020).

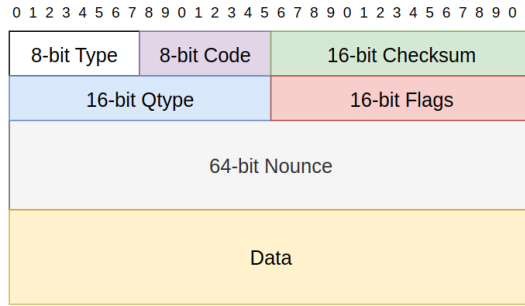
2.2.1 ICMPv6

The IP protocol provides a message protocol for network management that is available in both versions IPv4 and IPv6. Internet Control Message Protocol (ICMP) is part of the Internet protocol suite, and these messages are typically used for diagnostic or control purposes, or generated in response to errors. As not all ICMPs messages are deprecated and with no clear propose, they can be used to carry traffic information between domains and help to implement QoS.

The ICMPv6 messages 139 (Node Information (NI) Query) and 140 (NI Reply) were introduced in RFC 4620 with their applicability limited to debugging, network diagnostics, and network management (SHORE; PIGNATARO, 2014). According to that RFC, these messages can be used to discover the addresses and names of nodes on the other end of a point-to-point link. Figure 4 illustrates the structure of these packets. ICMPv6 messages 139 and 140 are encapsulated in an User Datagram Protocol (UDP) packet and include several useful fields that can be utilized by QoS provisioning strategies. Although ICMPv6 is not intended as transport for other protocols, it effectively serves the purpose of carrying network control data such as flow information.

In these messages, the field Type specifies if it is an ICMPv6 139 NI Query or 140 NI Reply message. The Code is an 8-bits field that has defined values, but for the FLOWPRI-SDN, new codes are defined once it would not be analyzed by conventional-infrastructure networks. The same occurs to the Qtype field that defines the type of information requested in a Query or supplied in a Reply. The Nonce MUST be a random or good pseudo-random value to foil spoofed replies. Nonetheless, such processes

Figure 4 – ICMPv6 139 and 140: Node Information Query and Reply.



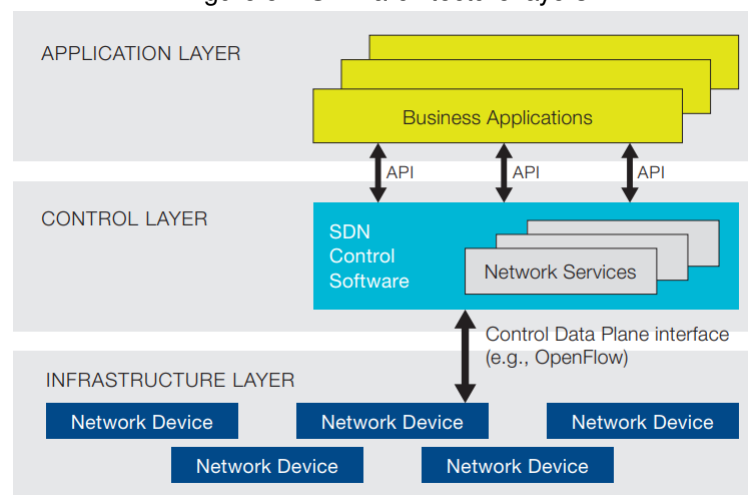
Source: (CRAWFORD; HABERMAN, 2006).

MUST check the received Nounce and ignore extraneous Replies. And at last, the Data field is where the control or traffic flow information can be encapsulated.

2.3 SOFTWARE-DEFINED NETWORKING

The conventional network architecture has a rigid structure with low programmability, which hinders the development of dynamic and collaborative solutions. However, the SDN architecture proposes to separate the data plane from the control plane by centralizing the control plane on a device called a controller (ONF, 2012). The controller has a global view of the network topology and can dynamically configure each of the data plane devices. This approach enables dynamic control, modification, and management of network behavior by means of software interfaces.

Figure 5 – SDN architecture layers.



Source: (ONF, 2012).

Figure 5 presents the main components of the SDN architecture, that are decomposed below:

- Application Layer: This layer consists of one or multiple network applications that utilize the controller’s functionality to achieve network management functions.

- **Control Layer:** The control layer consists of one or more programmable SDN controllers, responsible for monitoring and managing network traffic flow. These controllers have interfaces that connect them with other layers, enabling comprehensive management capabilities.
- **Infrastructure/Data Layer:** This layer consists of the infrastructure plan comprising network devices such as physical/virtual switches and access points. These devices are managed by the SDN controllers.

The SDN architecture enables exploring the network's programmability by means of controllers, thereby avoiding physical reconfigurations in the network infrastructure and reducing the time required to implement improvements. In a traditional network, to add new traffic flow treatment policies, each router would need to be reconfigured. However, in networks compatible with the SDN architecture, controllers distributed across the network can be programmed to instruct the devices along the flow path and configure them with the new policy only when necessary (Krinsky; Dvornik; Krinsky, 2019). The work of (Belgaum et al., 2019), cites some of the functionalities that can be employed with SDN:

- **Routing:** It's possible to determine routes that are different from traditional ones and making it dynamically.
- **Load Balancing:** Based on specific metrics, packets can be routed to another cluster of the same service.
- **Resource Management:** Statistical information can be collected, and based on that, resources such as bandwidth can be managed.
- **Green Computing:** Devices energy consumption can be controlled, for instance, by managing their need to remain on when there is no data flow.

2.3.1 Controller

The controller is the most crucial component of the SDN architecture because of the management and control it exercises over packet-switching equipment (Paliwal; Shrimankar; Tembhrne, 2018). Using the OpenFlow communication protocol, the controller can manage various elements of switches through message exchanges (TEAM, 2014).

With this protocol, the controller can add, update, and delete flow rule entries in the switches' internal tables, both reactively and proactively (ONF, 2012). In reactive mode, a switch requests action from the controller for a packet when there is no forwarding rule that matches the packet, forwarding that packet to the controller (packet-in

event). The controller then examines the packet headers and its data to make a decision. Then, it configures the switch with new forwarding rules to attend the data flow. In proactive mode, the processing and installation of new rules occur in a pre-scheduled manner, without the need for prior interaction with the switches regarding specific types of data packets.

One well-known controller is the RYU controller. As described in (Selvaraj; Nagarajan, 2017), it is written entirely in Python and is available under the Apache 2.0 license. Among the various protocols supported for managing network devices are OpenFlow, versions 1.0 through 1.5, Netconf (a protocol for network management and configuration), and OF-Config (a set of configurations for controller-switch communication).

2.3.2 Open Flow

The OpenFlow protocol was the first communication interface developed for the SDN architecture and it is also referred to as the Southbound Application Programming Interface (API) (ONF, 2012). Switches and controllers communicate through the exchange of OpenFlow messages, using a conventional socket interface. This protocol can be implemented on both real network devices and on software-based SDN controllers and switches (like Open Virtual Switch (OVS)).

Each version of the OpenFlow protocol has introduced new functionalities and changes compared to previous versions (KARAKUS; DURRESI, 2017). Version 1.0 allowed an output port to implement queue management, with multiple queues per port. However, the version 1.3 introduced a more distinct functionality known as Meter Tables. This feature enables the measurement of flows and the application of actions based on these measurements, allowing specific behaviors to be applied to each data packet flows.

2.3.2.1 OpenFlow Tables

OpenFlow Tables store flow forwarding rules for a datapath (switch) in a specific database (Open Virtual Switch Database (OVSDB) for the OVS, or in memory for commercial off-the-shelf hardware). As explained in (Selvaraj; Nagarajan, 2017), when a packet arrives at a switch, the header fields are analyzed to find a match in the installed Flow Table rules. Typically, switches can perform three basic actions: forward the packet according to the matched rule, discard it, or send it to the controller. However, it's also possible to implement more sophisticated actions, such as modifying packet header fields in this process. Figure 6 illustrates an OpenFlow forwarding table, its match and action fields.

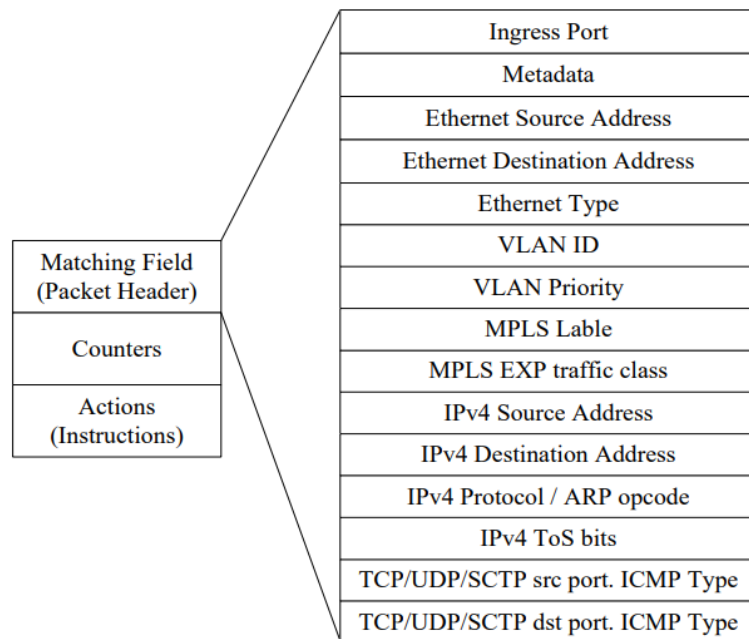
Figure 6 – OpenFlow forwarding table.

OpenFlow-enabled Network Device							
<i>Flow Table comparable to an instruction set</i>							
MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:..	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

Source: ((ONF), 2012).

The flow rules installed in these tables can utilize various fields to match packets targeted by defined actions. The most common match fields are source and destination IP addresses, protocol and, source and destination ports. The other match fields available are depicted in Figure 7. The counters are categorized into: Flow Table counters, flow entry counters, per-port counters, per-queue counters, per-group counters, and per-meter counters. Lastly, the instructions are sets of actions associated with flow rules, executed when a packet matches a match field. Instruction sets support various actions, including:

Figure 7 – OpenFlow struct rules.



Source: (BHOLEBAWA; DALAL, 2016).

- Setting the packet’s output port.

- Defining the queue associated with the output port where the packet will be placed.
- Dropping the packet.
- Adding or removing tags, such as Virtual Local Area Network (VLAN) or Multiprotocol Label Switching (MPLS).
- Associating a group so that the packet is processed according to the rules of that group.

Flow rules can be configured with cookies, timeouts and with actions to send packets to different queues in an output port. Additionally, a multi-table logic can be employed to route packets between rules in these tables until a final forwarding rule is reached. As a result, rules can act in a chain effect, and multiple actions can be executed on the same data flow ¹.

2.3.2.2 Meter Table

A Meter Table consists of meters associated with flows (ONF, 2012). Figure 8 presents the meter table's fields. These meters measure a type of rate on packets, specified in their definition, and enable controlling the flow of packets according to the defined rate. This component allows implementing several QoS operations, such as limiting flow transfer rates, and can be combined with queue strategies on switch output ports.

Meters are associated with actions in flow rules. Therefore, multiple meters can be applied to the same flow through successive Flow Tables. Each entry in the Meter Table is identified by its meter Identification (ID) and contains the following fields:

- Meter identifier: A 32-bit unsigned integer that uniquely identifies the meter.
- Meter bands: A list of meter bands, which measure a specific component and specify how packets should be treated when processed.
- Counters: They update when packets are processed by a meter.

Figure 8 – Meter table.

Meter Identifier	Meter Bands	Counters
------------------	-------------	----------

Source: (ONF, 2012).

Each meter band rule also contains:

¹ Data flow: Set of packets that satisfy a flow rule

- Band type: Defines how packets are processed.
- Rate: Defines the rate limit of forwarding that should be applied to the flow.
- Counters: They are incremented when a packet is processed.
- Specific type arguments: Some band types support optional arguments, such as packet dropping if the flow exceeds the defined rate.

Figure 9 shows the fields for the Meter Bands. These rules are used to measure the maximum delivery rate of a packet flow (ONF, 2012). Therefore, flows that exceed the limit set by one meter band rule are adjusted by discarding or remarking packets.

Figure 9 – Meter bands.

Band Type	Rate	Counters	Type specific arguments
-----------	------	----------	-------------------------

Source: (ONF, 2012).

Therefore, Meter Table rules are essential in network resource management mechanisms, as they allow shaping of flows at desired rates. By combining this component with HTB priority queues and bandwidth allocation models, it is possible to meet data traffic QoS and prioritization.

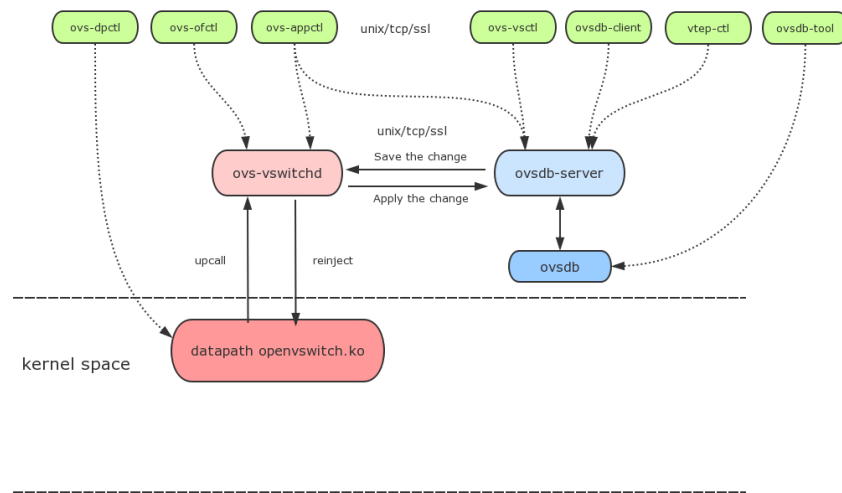
2.3.3 Open vSwitch

A virtual switch that supports the OpenFlow protocol is OVS, whose architecture can be seen in Figure 10. This component consists of a set of Flow Tables, group tables, and an external communication channel that connects to a controller (ONF, 2012). OVS also supports HTB queues to shape and separate traffic. The main components of OVS are:

- ovs-vswitchd: It is a daemon interface that implements the virtual switch. This module utilizes the operating system's kernel to access network interfaces and perform flow-based switching, effectively implementing the virtual switch.
- ovssdb-server: It is a database server that stores all configurations of the virtual switch. The ovs-vswitchd tool queries this server to obtain the switch configuration.
- ovs-dpctl: It is a tool for configuring the switch's datapath module, which operates in kernel mode.
- ovs-vsctl: This is a tool used to query and update the configurations of ovs-vswitchd, via a connection to an ovssdb-server.

- `ovs-appctl`: This tool provides a straightforward way to invoke specific commands via the command line for `ovs-vswitchd`.
- `ovs-ofctl`: It is a utility for querying and controlling OpenFlow switches and controllers.

Figure 10 – OVS architecture.



Source: (SOBYTE, 2022)

The `ovs-vswitchd` works in conjunction with datapaths to implement flow-based packet switching (SOBYTE, 2022). This component communicates with controllers using the OpenFlow protocol and connects to the `ovsdb-server` database using the OVSDB protocol.

An `ovs-vswitchd` supports multiple independent datapaths. Upon initialization, `ovs-vswitchd` reads information from the `ovsdb-server` to configure itself and its datapaths. When a modification occurs in the OVSDB, `ovs-vswitchd` is notified and updates its configurations accordingly.

The datapath that utilizes the kernel module listens for packets from a network interface. In the kernel space, memory allocation is limited, so only a few flow rules can be stored. If the datapath doesn't find a match in the flow rule cache in the kernel space, it forwards the packets to `ovs-vswitchd`, which operates in the user space.

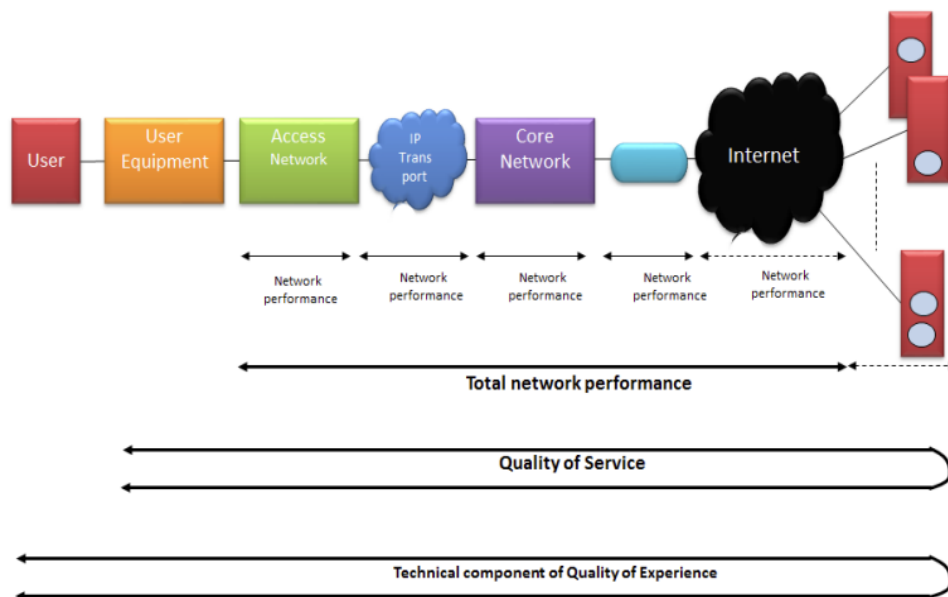
The `ovs-vswitchd` maintains all flow rule entries from datapaths that have been loaded from the database. Therefore, `ovs-vswitchd` updates the Flow Table in the kernel space with the corresponding rule. If there isn't a matching rule, it proceeds to forward the packets to the controller and waits for a new rule. Once a new rule is received, it updates the table in the kernel space. Subsequently, `ovs-vswitchd` reinjects the packets into the datapath. Finally, the datapath forwards or discards packets based on the cached Flow Table entries for a network interface.

2.4 QUALITY OF SERVICE

As explained in previous sections, SDN has the tools to implement more powerful QoS strategies. In the current legacy network architecture, application flows contend for available resources using best-effort delivery mechanisms provided by transport protocols. On the Internet, each AS manages traffic and routing policies within its domain, making it difficult to establish agreements to guarantee QoS on a per-flow basis for conventional clients. This way, it is expected that the popularity and variety of multimedia-rich applications along with the increased number of users has led to an ever-increasing pressure on the underlying networks, motivating a shift to the SDN network paradigm and new QoS strategies (AL-JAWAD et al., 2021).

In this context, QoS can be defined by the characteristics of a telecommunications service that ensure the ability to meet the needs of the service user, both stated and implied requirements (JANEVSKI; JANKOVIC; MARKUS, 2017). The primary objective of QoS is to prioritize services, which includes providing dedicated bandwidth, controlling jitter and latency (necessary for real-time and interactive traffic), and avoiding packet loss characteristics. Furthermore, it is crucial to ensure that prioritizing one or more flows does not adversely affect other flows (CACHEDA et al., 2007). Figure 11 illustrates the factors that impact QoS provided for an application data flow. Consequently, at each network hop, the data flow is influenced by several QoS Key Performance Indicators (KPI)s, summarized as follows:

Figure 11 – QoS and QoE in a network.



Source: (JANEVSKI; JANKOVIC; MARKUS, 2017).

- Bandwidth: The maximum number of bits that a transmission path can carry.

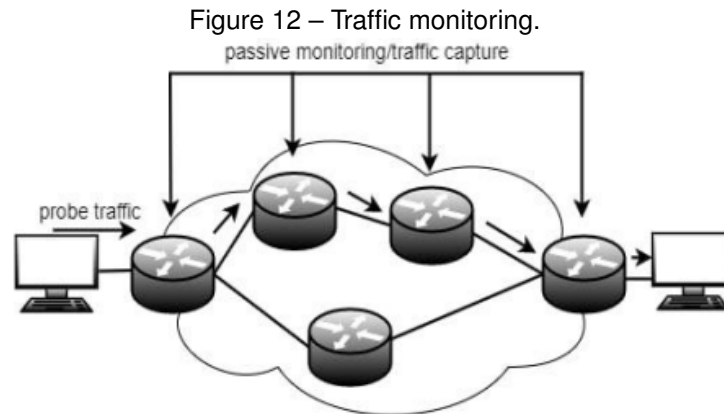
- Propagation delay: The time that a packet requires, as a function of the combined length of all transmission paths and the speed of light through the transmission path.
- Queuing delay: The time that a packet waits before being transmitted.
- Jitter: Both the average delay and variability of delay (jitter) matter, since the two together establish a confidence interval for how long a packet can expect to reach its destination.
- Packet loss: The probability that a packet never reaches its destination.

2.4.1 Traffic Monitoring

In order to keep the QoS levels for each application traffic flow, network operators have to provide Real-time network monitoring, collect data from network forwarding devices and data flow transmission states, and analyze network performance indicators and metrics. The monitoring system should be accurate, easy to use, and quick enough to reflect network performance in a timely way without compromising the network performance (ALKENANI; NASSAR, 2022).

The monitoring techniques can be passive, active, or hybrid. Figure 12 illustrates how traffic can be monitored using passive methods, and how active methods inject traffic to measure the network state. Passive methods do not inject or alter traffic data to generate performance statistics. They rely on collecting and processing original traffic data to estimate network characteristics in a non-intrusive manner. Some forwarding devices include built-in tools that provide highly aggregated data and minimal information about network conditions or traffic patterns, which administrators can query. Alternatively, traffic data can be recorded from forwarding devices by duplicating it to another device that maintains the traffic database. In SDN, rules can have multiple actions, allowing the controller to passively analyze traffic, which is simpler than in legacy networks (ALKENANI; NASSAR, 2022).

Active measurement methods, on the other hand, are based on injecting additional traffic (such as probe packets) between endpoints in the network and observing their behavior to obtain information about network properties. This technique can provide reliable data and estimate the quality of observed network segments, whether on large-scale networks, end-to-end, or per link, measured at the destination. However, active measures are network-invasive; probe packets can degrade network performance and affect the collected information. For finer-grained, highly flexible, and timely network performance verification, more sophisticated monitoring techniques are required. Due to the ongoing nature of network performance verification, active moni-



Source: (ALKENANI; NASSAR, 2022).

toring techniques may not be the most feasible approach for this purpose (ALKENANI; NASSAR, 2022; HAXHIBEQIRI et al., 2021).

Hybrid Monitoring measurement refers to active and passive measurements in conjunction with one another. In these methods, probe packets are sent to the network and their progress in the network is then monitored with passive traffic collection. This enables to measure both end-to-end and intermediate link delays (ERVASTI, 2016).

2.4.2 Applications over the Network

In IP networks, the application data traffic flowing can be carried by two transport protocols: UDP and Transmission Control Protocol (TCP). The first protocol, is not connection oriented, it will send packets to the destination without taking care if they are being received and doesn't provide congestion avoidance methods. On the other hand, TCP is a connection oriented protocol, it means that the sender and the receiver sync the packets sent and received, so if a packet is not confirmed it will be resent. Also, TCP has a fair share congestion protocol to avoid congestion. This way, applications that are sent by UDP can send traffic with less delay, but with the chance of losing some packets, whereas, applications that send packets with TCP can have traffic rate limited, but without congestion and with reliable transfers (BASUMALLICK, 2022).

Above the transport layer, is the application layer. This layer has many protocols that set of standards and rules to the communication between end-user applications over the network. Specific services and functionality are provided by these protocols to support various types of application-level communication, such as file transfers, email, remote terminal connections, and web browsing (PATIL, 2023).

Traffic from applications on the Internet consists mainly of multimedia applications, which combine text data, video, and audio. Data applications are distributed across the Internet in various forms, such as HTML pages, video/audio streaming, on-line games, chat, video and audio calls, and file downloads. Bandwidth requirements

for these applications vary greatly depending on the size of requested files, the type of application, and the data rate. Moreover, some applications may tolerate errors or packet loss, such as video streaming, while others, like file transfers, cannot tolerate any loss at all (KHANVILKAR et al., 2004).

These applications can be classified by their time dependency requirements. Some applications are time-dependent and the time at which the output is generated is important. It means that, the application is sensitive to delay and jitter, and must receive the data as fast as it can. On the other hand, non-real-time applications are those that doesn't have hard timing requirements, but they are still delay sensitive (CHEN; FARLEY; YE, 2004).

The applications can also be classified in symmetric or asymmetric data flows. Symmetric applications demand request and response data, like in video calling applications. These means that, the application needs equal resource consumption in both ways. Whereas, in asymmetric applications, the request is less resource consuming than the response, because it will get data at a higher rate than it will send. Table 1, summarizes many applications and their QoS metrics. These metrics are based on generic traffic studies made in past researches and from the specification provided in their websites (CHEN; FARLEY; YE, 2004; GOOGLE, 2025; COMMISSION, 2013; MICROSOFT, 2025).

The literature lacks in researches that explore the QoS requirements of specific applications. Nonetheless, it is possible to capture traffic and make inferences over the application flow requirements.

2.5 IDENTIFYING THE TRAFFIC FLOW REQUIREMENTS

Before implementing QoS and reserve resources to application flows, traffic has to be classified in application classes. In the literature, it is possible to find some methodologies to identify traffic into classes or groups based in some information, such as port-based model, Deep Packet Inspection (DPI), traffic statistics and using machine learning.

Port-based classification was one of the most widely used techniques in the past because many applications used fixed port numbers assigned by the Internet Assigned Numbers Authority (IANA) organization. However, over time, limitations of this approach began to emerge. Numerous applications do not have registered port numbers, and many use dynamic port negotiation mechanisms to evade firewalls and network security tools. Additionally, the use of IP layer encryption has made it impossible to discover the original port numbers (AMARAL et al., 2016).

The DPI technique, also known as the payload approach, was proposed to

Table 1 – QoS metrics for general network applications.

Application	Classification	Bandwidth	Delay	Loss
Web Browsing	Non Real Time and Asymmetric	30kbps	< 400ms	0
E-mail	Real Time and Asymmetric	<10Kbps	Low	0
FTP	Non Real Time and Asymmetric	High	Med	0
Text chat	Non Real Time and Symmetric	Low	< 200ms	0
Audio Broadcasting	Real Time and Asymmetric	60-80 Kbps	< 150ms	<0.1%
Video Broadcasting	Real Time and Asymmetric	HD MPEG-1: 1.5Mbps HD MPEG-2: 4 Mbps HD MPEG-4: 500Kbps	<150ms	<0.001%
Audio on Demand	Real Time and Asymmetric	MP3 128K MPEG-1: <500Kbps AAC MPEG-2: 384Kbps	<150ms	<0.1%
Video on Demand	Real Time and Asymmetric	HD MPEG-1: 1.5Mbps HD MPEG-2: 4M-60Mbps HD MPEG-4: 500Kbps	<100ms <50ms < 150ms	<0.001%
Audio Conferencing	Real Time and Symmetric	Many coding standards: 10K-80Kbps	<150ms	<1%
Video Conferencing	Real Time and Symmetric	Many coding standards: 80K-2Mbps	<150ms	<0.01%
Videophony	Real Time and Symmetric	Many coding standards: 80K-2Mbps	<100ms	<0.01%
VOIP	Real Time and Symmetric	Many coding standards: 10K-80Kbps	<150ms	<1%
Online Gaming	Real Time and Symmetric	2M-4Mbps	<150ms	<3%

Source: Adapted from (CHEN; FARLEY; YE, 2004; GOOGLE, 2025; COMMISSION, 2013; MICROSOFT, 2025).

overcome the limitations of port-based classification techniques. In this approach, the payload of the packets are compared to a series of signatures (SERAG et al., 2024). These signatures are regular expressions that are evaluated by an automaton sequentially. Therefore, one disadvantage of this approach is maintaining up-to-dated signatures, since the amount of applications keeps increasing with time and also packets can be encrypted, turning down the signature matching precision.

Another technique is to capture traffic streams and to analyze their statistical properties instead of the payload contents. In this approach, it is assumed that traffic with similar QoS requirements has comparable statistical features (SERAG et al., 2024). As a result, it is possible to classify flows into clusters with similar patterns using some properties, such as the size of the initial few packets and the packet arrival timings.

To overcome the limitations of traditional classification methods, Machine Learn-

ing (ML) algorithms are used. ML algorithms exhibit variations in their methodology, and it is possible to classify them into four distinct categories according to the nature of the data they handle, the output they generate, and the specific task or problem they aim to address: supervised learning, semi-supervised learning, unsupervised learning, and reinforcement learning (SERAG et al., 2024).

- **Supervised learning:** It constructs a mathematical model using a labelled training dataset that includes both inputs and their known outputs.
- **Unsupervised learning:** It is utilized for clustering and data-aggregation tasks, where the data provided to the learner are unlabeled. In such scenarios, algorithms group the data into distinct clusters based on similarities found in the feature values.
- **Semi-supervised learning:** In practical situations, the cost of labeling data can be very high, resulting in limited availability of labeled data. Semi-supervised learning algorithms can be an alternative. It consists of two stages: the initial step involves analyzing labeled data to generate a general rule, which is subsequently utilized to deduce unlabeled data.
- **Reinforcement learning:** It represents a form of ML training that relies on rewarding favorable behaviors and/or penalizing unfavorable ones. In general, this kind of agent has the ability to perceive and interpret its environment, take actions, and acquire knowledge through trial and error.

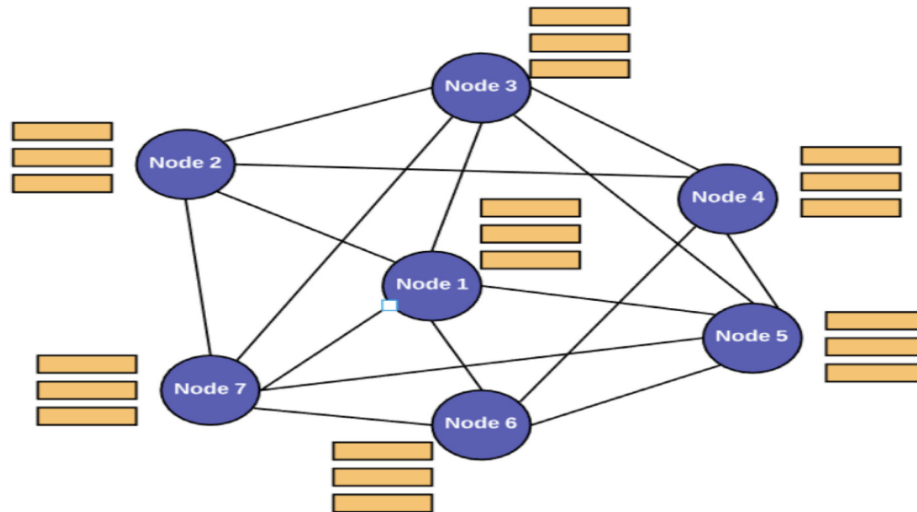
2.6 BLOCKCHAIN AND QoS TRUSTNESS

As first presented in the Introduction section, the log for the acQoS provided to a flow can be stored using blockchain. Integrating the blockchain technology in SDN networks can eliminate the need for centralized third-party entities in QoS-supported inter-AS routing models by exploiting blockchain's decentralized feature. Provisioning of QoS-based routing services across ASs poses grueling challenges due to the distributed nature and business confidentiality hesitations (KARAKUS; GULER; ULUDAG, 2021). For example, consensus algorithms could be applied to prove that the administrative domains wasted computational efforts to provide the necessary resources for the QoS level a flow requested. Thus, the flow information of each domain can be stored in a secure distributed database, such as a blockchain.

Blockchain technology is a decentralized and peer-to-peer communication driven (MOHANTA et al., 2019). Blockchain stores information permanently across a network among nodes in a decentralized and distributed way. Figure 13 shows a basic

blockchain architecture. Each node in the network can store the local copy of the Blockchain system, which is periodically updated to have the consistency among all nodes. It enables nodes that do not trust each other to keep a consistent database. A node can perform different activities like initiate a transaction, validate a transaction or perform mining (data validation). Then, the blockchain collects transaction information and enters it into a block.

Figure 13 – Blockchain architecture.



Source: (MOHANTA et al., 2019).

Consensus may be implemented in different ways, such as through the use of lottery-based algorithms including Proof of Elapsed Time (PoET) and Proof of Work (PoW) or through the use of voting-based methods including Practical Byzantine Fault Tolerance (PBFT) and Paxos (HYPERLEDGER, 2017). Each of these approaches targets different network requirements and fault tolerance models.

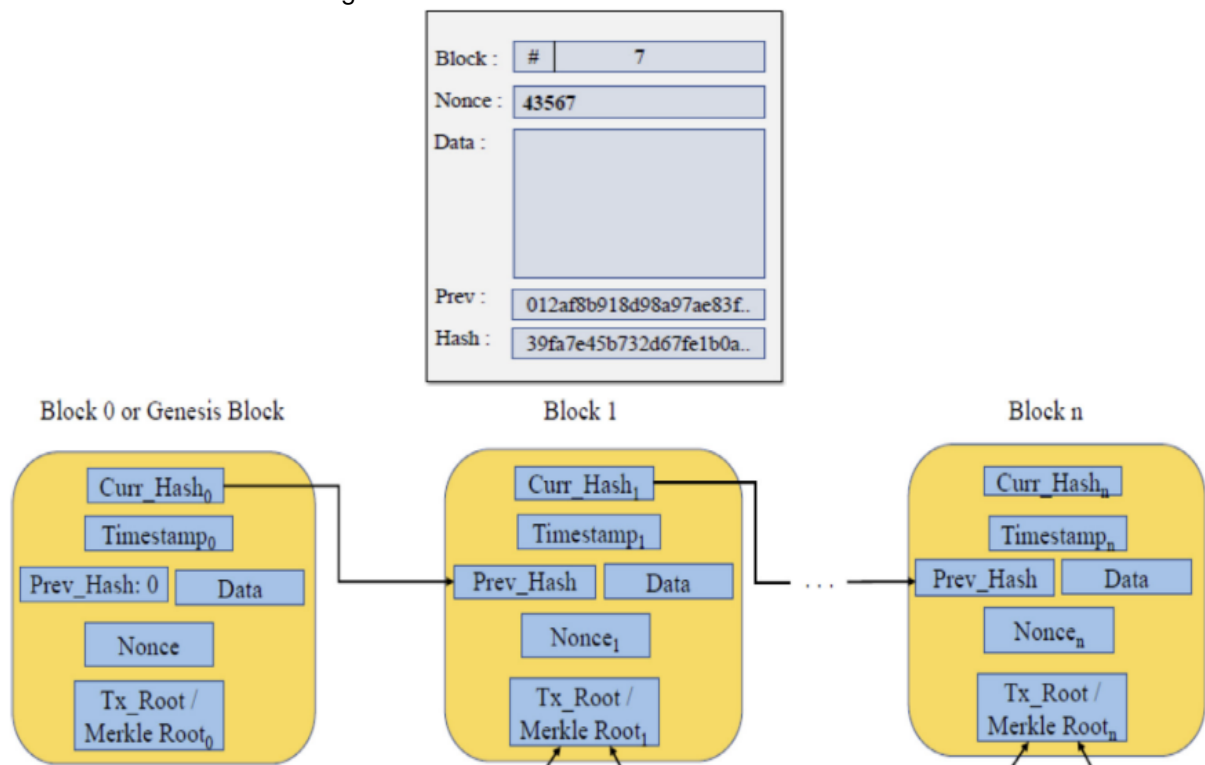
The lottery-based algorithms have advantage in large scale number of nodes, since only the winner of the lottery proposes a block and transmits it to the rest of the network for validation (HYPERLEDGER, 2017). On the other hand, these algorithms may lead to forking when two “winners” propose a block. Each fork must be resolved, which results in a longer processing time.

The voting-based algorithms are advantageous since they provide low-latency finality. When a majority of nodes validate a transaction or block, consensus exists and finality occurs (HYPERLEDGER, 2017). Because voting-based algorithms typically require nodes to transfer messages to each other in the network, the more nodes that exist in the network, the more time it takes to reach consensus. This results in a trade-off between scalability and speed.

Figure 14 shows the structure of blockchain technology. It involves (MOHANTA et al., 2019):

- Block: It is the collection of valid transactions. Once a transaction is validated, it is added to the existing Blockchain by means of its persistence on a block.
- Block ID: Block identifier inside the blockchain.
- Nounce: Unique identifier, number only used once.
- Data: Data to be stored in the block.
- Previous Block hash: Every block inherits from the previous block. Blockchain system process previous blocks hash to create the new blocks hash, making the Blockchain tamperproof.
- Merkle Tree Root: The transactions are organized in a Merkle Tree structure. The root of the Merkle tree is a verification of all the transactions.

Figure 14 – The blocks of a blockchain structure.



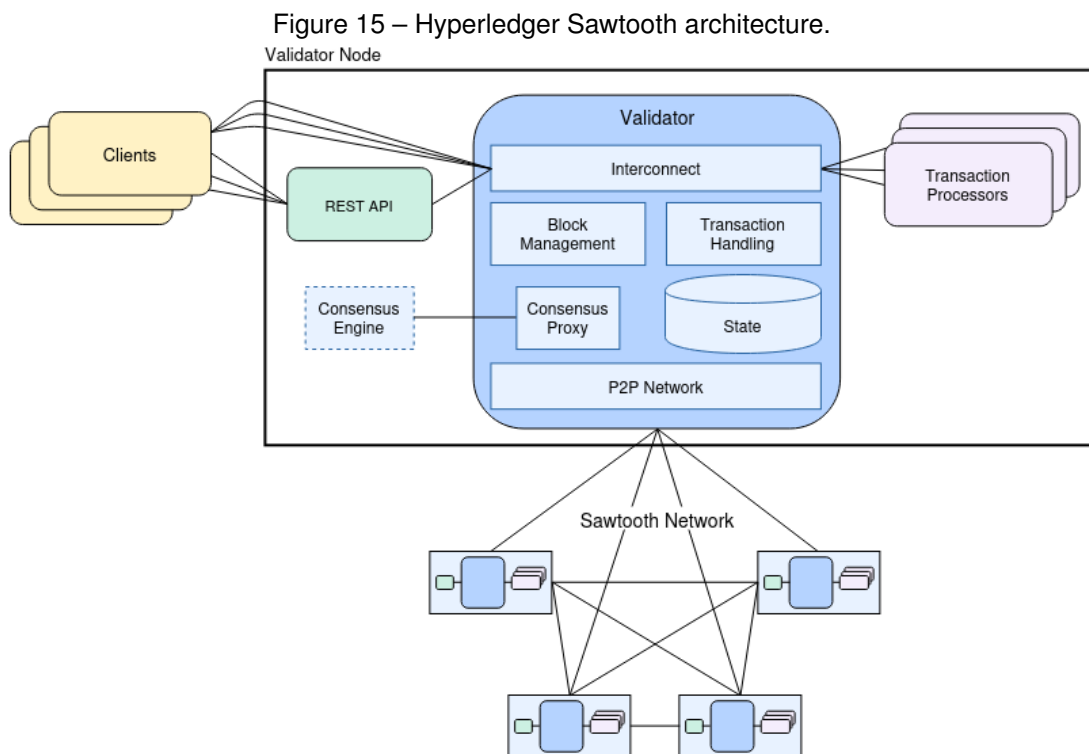
Source: (MOHANTA et al., 2019)

2.6.1 Hyperledger Sawtooth Framework

Hyperledger Sawtooth is an open source blockchain platform for building distributed ledger applications and networks. The design philosophy targets keeping ledgers distributed and making smart contracts safe, particularly for enterprise use (FOUNDATION, 2024). However, the main aspect of Sawtooth platform is that it makes possible

to develop and deploy an application by providing a clear separation between the application level and the core system level.

The Sawtooth architecture is shown in the Figure 15. The main components of a Sawtooth node is the validator, that connects the other components and handles the peering with other nodes in the Sawtooth network. The transaction processor is the component that rules the business logic. It defines the block payload and how the transactions are stored. The block validation process is handled by the consensus engine component. Then, the Rest API component is the interface between the client application and the validator protobuf protocol that handles the transactions in batches.



Source: (FOUNDATION, 2024).

Each application defines the custom transaction processors for its unique requirements. Sawtooth provides several example transaction families to serve as models for low-level functions (such as maintaining chain-wide settings and storing on-chain permissions) and for specific applications such as performance analysis and storing block information. Also, there are examples in many languages, to help shortening the learning curve of the platform.

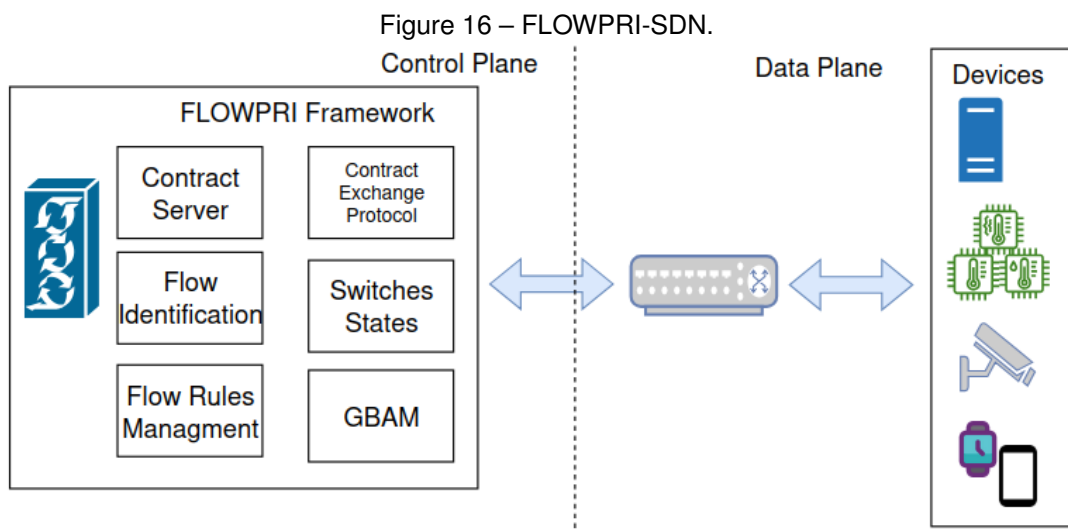
2.7 FLOWPRI-SDN FRAMEWORK

Flowing with priority in SDN (FLOWPRI-SDN) is a SDN framework for per-flow QoS provisioning that can cover multiple administrative domains (AS, ISPs or other

entities) (JÚNIOR; FIORESE, 2023). The FLOWPRI-SDN is a Python application that leverages the RYU controller base and the OpenFlow 1.3 protocol.

The RYU controller version 4.34 has several network management functions implemented that can be used. It also supports several protocols to manage network devices, such as OpenFlow from versions 1.0 to 1.5, Netconf and OF-config. Figure 16 shows the FLOWPRI-SDN architecture, and its main components are presented as follows:

- Contract Server: Module responsible for receiving QoS contract requests.
- Flow identification: Module for computing the DSCP code of the correspondent QoS contract of a flow.
- Flow rules management: Module to store the active rules of each switch.
- Switch States: Module responsible to manage the bandwidth allocated and free of each class of service (queue).
- G-BAM: Module responsible to allocate the available bandwidth between all switches of a domain that are in the route of a flow.
- Contract exchange protocol: The module responsible to handle the announcement of the contracts. It encapsulates the contract in ICMPv4 messages to send to other domains in the route of a flow. Also, it answers for contract requests of other FLOWPRI-SDN frameworks.



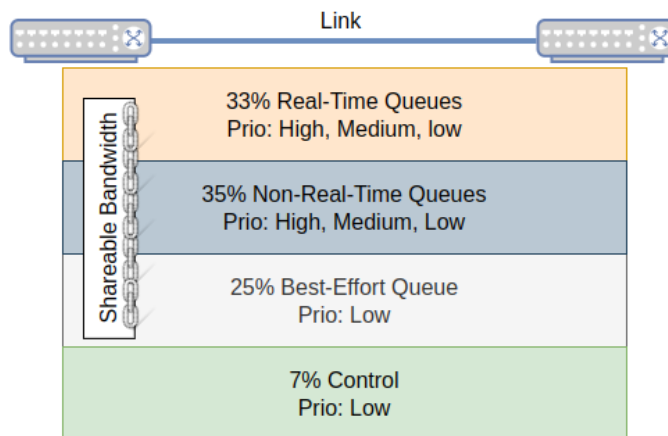
Source: (JÚNIOR; FIORESE, 2023).

This framework receives QoS requests from hosts in the format of contracts. Then, it synchronizes the flow requirements along the next hop domains, to provide multi-domain QoS. No routing mechanism is designed at this point. It is assumed

that all the routing strategies comes from an external API or is manually informed to FLOWPRI-SDN previously. Also, the traffic must flow from one host to another through the same route (static).

In each switch, the FLOWPRI-SDN divides the link bandwidth into four classes using HTB queues: control (7%), best-effort (25%), real-time (33%) and non-real-time (35%), as seen in Figure 17. With that, it can limit the traffic usage of QoS flows with OpenFlow meter rules and set the right queue to forward packets. In each QoS class, the bandwidth is shared by three leaf HTB queues. The G-BAM mechanism controls the bandwidth sharing for QoS flows inside the borrowings between QoS classes. The HTB mechanism leverage the bandwidth that is not used from other queues to the best-effort queue, in a non-reserved way. Tables 2 and 3, present bandwidth values available for reservation and the respective DSCP code that will be used to mark the flow packets.

Figure 17 – Link configuration.



Source: (JÚNIOR; FIORESE, 2023).

Table 2 – DSCP codes for real-time flows.

Banda	Prioridade 1 DSCP:Fila	Prioridade 2 DSCP:Fila	Prioridade 3 DSCP:Fila
32kbps	1:0	11:1	21:2
64kbps	2:0	12:1	22:2
128kbps	3:0	13:1	23:2
500kbps	4:0	14:1	24:2
1Mbps	5:0	15:1	25:2
2Mbps	6:0	16:1	26:2
5Mbps	7:0	17:1	27:2
10Mbps	8:0	18:1	28:2
25Mbps	9:0	19:1	29:2

Table 3 – DSCP codes for non-real-time flows.

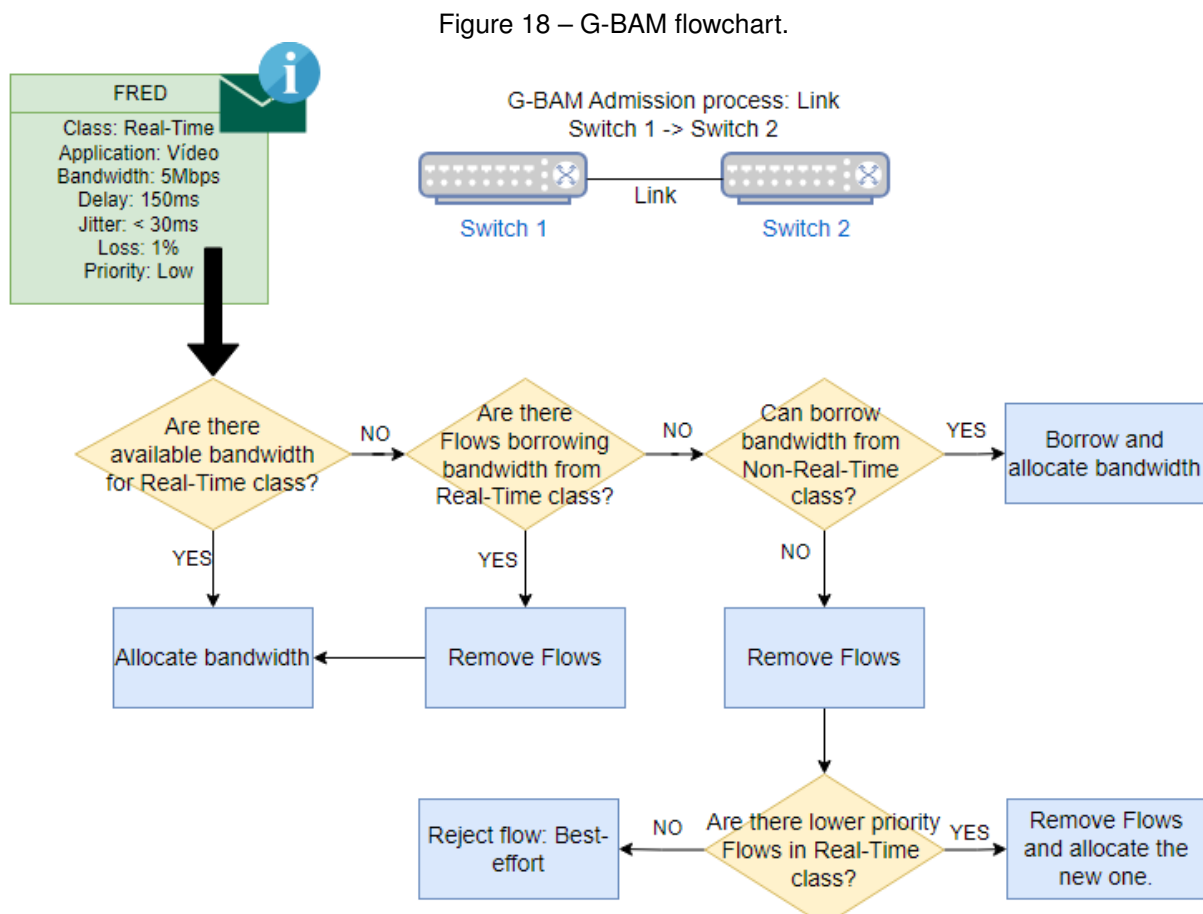
Banda	Prioridade 1 DSCP:Fila	Prioridade 2 DSCP:Fila	Prioridade 3 DSCP:Fila
32kbps	31:3	41:4	51:5
64kbps	32:3	42:4	52:5
128kbps	33:3	43:4	53:5
500kbps	34:3	44:4	54:5
1Mbps	35:3	45:4	55:5
2Mbps	36:3	46:4	56:5
5Mbps	37:3	47:4	57:5
10Mbps	38:3	48:4	58:5
25Mbps	39:3	49:4	59:5

In short, the first step for a host to request QoS for a particular traffic flow is by sending its requirements to the nearest domain with a FLOWPRI-SDN framework (edge domain). This contract must inform the minimum requirements about bandwidth, packet loss, jitter, and delay. Then, the controller is ready to identify the data flow as

soon as the first packet arrives with the packet-in OpenFlow event. So, it executes the G-BAM module to accept the flow and creates the QoS rules (the forwarding and the metering rules). These rules are created in the switch tables and stored in the switch instances within the FLOWPRI-SDN.

2.7.1 Generalized Bandwidth Allocation Method

The Generalized Bandwidth Allocation Method (G-BAM) method is responsible for flow admission, prioritizing bandwidth allocation to ensure that the highest priority flows are accommodated within their allocated limits for each service class. If necessary, flows can borrow bandwidth from others (JÚNIOR; FIORESE, 2023). For best-effort flows, only forwarding rules are configured in the switches, directing packets to the best-effort queue. In this case, no information is stored in the FLOWPRI-SDN switch manager. However, for QoS flows, FLOWPRI-SDN employs the G-BAM method to assess each switch along the route to ensure compliance with bandwidth allocation policies. Figure 18 presents the G-BAM method.



Source: Adapted from (JÚNIOR; FIORESE, 2023).

Traffic flows with higher priority will have preference and can remove less priority flows if needed. Therefore, first the G-BAM tries to allocate bandwidth in the class of

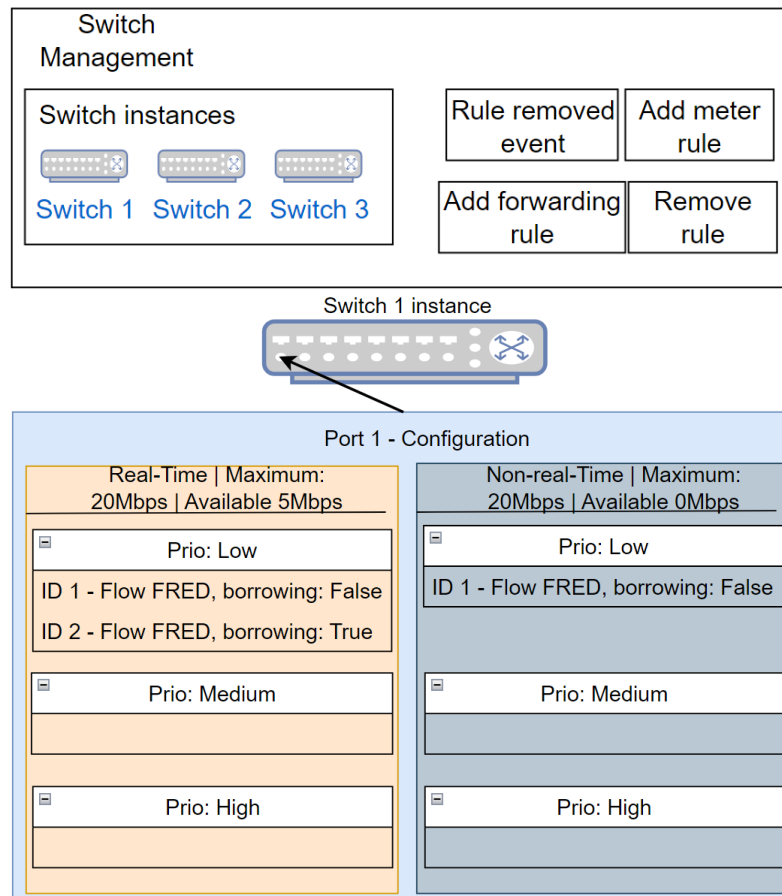
service specified in the QoS contract. If there is enough bandwidth, it creates the QoS rules in the switches' forwarding tables, as shown in Figure 19 and, it stores a copy in the switch instance within the FLOWPRI-SDN framework, as depicted in Figure 20.

Figure 19 – QoS rules.

```
OFPST_METER_CONFIG reply (OF1.3) (xid=0x2):
meter=14 kbps bands=
type=drop rate=5000
table=2,ip,nw_src=172.16.10.1,nw_dst=172.16.10.4,nw_tos=28 actions=meter:14,set_queue:3,output:4
```

Source: The Author, 2024.

Figure 20 – Switch management module.



Source: The Author, 2024.

If there is insufficient available bandwidth, the G-BAM first checks if there are flows within the same service class that are borrowing bandwidth. These flows are removed to make space for the new flow. If no flows are borrowing bandwidth, the G-BAM attempts to borrow bandwidth from the other QoS class, marking the rules with a "borrowing bandwidth" label. If this attempt fails, the last resort is to remove lower priority flows from their original classes to reserve the necessary resources. If none of these steps are successful, the flow cannot be accepted, and it is informed via ICMPv4.

The forwarding rules are configured with an idle timeout of 2 seconds and a hard timeout of 5 seconds. When a flow expires due to idle timeout, it indicates that the flow has ended, and the rules must be removed from the switches and instances by triggering the flow removal event, and freeing bandwidth to the correspondent class (queue). However, if a flow rule expires due to hard timeout, the FLOWPRI-SDN find the correspondent QoS contract and the QoS rules are configured again using G-BAM.

3 RELATED WORK

In this chapter, the related work is presented. First, we present papers that implement resource reservation or routing algorithms based on bandwidth in SDN. Also, researches related with traffic classification are explored to compare their architecture and objectives. At last, the works related with blockchain implementation in the SDN scenarios are depicted, in order to analyze how trusting can affect QoS implementation in multi-domain SDN networks.

The searches were performed in the scientific databases as follows: Google Scholar, IEEE Explorer and Science Direct. Then, for each matter of search, it was formed a sentence combining the words blockchain, traffic classification and QoS with: Internet, framework and SDN. This way, the papers were sorted from most recent to least recent to be inspected. After reading, we choose the following papers to represent the state of the art of the QoS provisioning scenario.

3.1 QOS IN SDN

The work presented in (AZIZ et al., 2023), provides a framework that can help network service providers to allocate network resources dynamically and proactively, to minimize the impact of network congestion for delay-sensitive applications. The framework works capturing the network traffic using port mirroring technique. Then sets up the database to store the captured network traffic and categorize the captured network traffic into one of six QoS classes proposed using DPI technique. This way, it can predict the traffic load in each QoS class using the proposed RNN-BLSTM neural network model. Then, the framework allocates the network resources based on the traffic priority and the predicted traffic volume, constrained by the available data rate, using the designed algorithm. The results obtained showed that the framework can predict the network traffic in each QoS class for better resource utilization with average accuracy of 97.68%.

QoSFlow is a QoS management solution for SDN networks. It enables QoS control using a flow-based approach (BARKIRE et al., 2025). Users specify the maximum allowed bandwidth for particular high-level applications in the home network using a web configuration tool. This tool then creates the rate shaping configuration, which has two main components, namely the flow classifier and the SDN-based rate controller. The flow classifier contains a lookup table where the key is a flow tuple composed of the source IP address, destination IP address, protocol, source port, and destination port. This component uses two modules to perform traffic classification.

The first module performs early identification of HTTP and HTTPS traffic. The second component, the SDN-based rate controller, performs identification of other flows. However, it was a preliminary study and will not be included in the comparative table.

The paper published on (AL-JAWAD et al., 2021) introduces an innovative Reinforcement Learning-based framework for multimedia-based SDN environments that selects the most appropriate routing algorithm from a set of centralized routing algorithms to maximize the return reward from the network and enable QoS provisioning. Four routing algorithms are adopted, Minimum Hop Algorithm, Shortest Widest Path, Widest Shortest Path and Minimum Interference Routing Algorithm. In this methodology, the controller receives a packet-in message from a newly incoming request. Based on the resources available, the controller decides whether the request is accepted or not. If the request is accepted, the traffic type is identified first and then the most suitable routing algorithm for the specific service type is selected. For the QoS-based traffic, live HD video streaming is employed. While background traffic is represented by buffered SD video streaming. The applications are identified based on the port number. The results show that the proposed framework outperforms the routing algorithms alone and finds the best trade-off between throughput, packet loss and rejection rate for the QoS-based traffic class without penalizing the other traffic classes.

In (THAZIN; NWE; ISHIBASHI, 2019), is proposed an end-to-end dynamic bandwidth allocation scheme based on QoS demand in SDN and efficient use of network resources by providing network resources to network traffic according to their QoS demands and the current network status. When a user wants a desired QoS such as end-to-end delay, the user can request the controller by sending a request packet which includes the flow information such as the source, destination and the required QoS factors such as the amount of bandwidth they need and a delay tolerance value. For the delay demand flow, they first choose the minimum hop count paths in order to minimize the flow delay when the link bandwidth is enough for it. Otherwise, they find the alternative path with the least loaded. For bandwidth allocation, three different queues are set at each port and assign different priorities to them. Therefore, one of the three queue priorities can be assigned for the different QoS flows, three QoS priority classes (high, medium, and low). The results were obtained making a comparison between the proposed scheme, the Traditional Single Path routing scheme, and Flow-Based Multipath routing scheme. The proposed scheme provides congestion management to detect the bottleneck link and reroute the highest priority flow to other paths to mitigate the QoS service degradation of priority flows. The proposed scheme provides the highest average throughput than the other two schemes.

The architecture published in (OLIVEIRA et al., 2021), presents a generic architecture that implements the “RSVP” Requester and “RSVP” Provider modules to

provide QoS for sensitive entities. In the model, a host requests a service to the provider with its QoS specifications. The provider then checks if the network can transmit the flow following the requirements, otherwise it informs the QoS provisioning forecast. After that, the controller queries the necessary resources on the forwarding devices according to the previously established service category. In case of success, service will deliver its data with QoS QoS queues that controller previously configured in forwarding devices. Otherwise, the data packets will be treated as the other non-priority flows. Simple tests are performed to evaluate the proposed architecture, evolving one sensitive flow against some non-sensitive ones. This way, the obtained results shows that the methodology is more efficient in scenarios where big files are exchanged in the network, when in the other cases the conventional non-QoS approach seemed to over-perform the model.

The work proposed in (GHALWASH; HUANG, 2018), presents a framework for monitoring, route determination, rule preparation, and configuration functionalities. The monitoring module analyzes ports utilization and probs the links delay. The route determination module relies on the shortest path algorithm, with or without QoS guarantee. Two QoS parameters, namely, port utilization and delay, are considered in the monitoring and the route determination. Results showed that using the presented framework, with or without QoS reduces the overall average delay by 57%, jitter by 25% and packet loss by 67%. Moreover, the monitored port utilization was reduced by 30% on average.

In (JOSHI; KATAOKA, 2019), the authors present a privacy preserving E2E QoS mechanism in multi-domain SDN. They defend that, guaranteed E2E SDN needs complete information about the network. In multi-domain SDN, information sharing among different operators mainly have privacy and security issues; thus, operators may not share such network information. The main objective of the paper was to measure the amount of private information preserved in the domains. In PRIME-Q, only the binary decision about the domain networks are reported at a single point to take the final decision about E2E QoS. Therefore, the controllers keep track of maximum available resources by monitoring and periodically updating the network resource information. When a request comes at the source domain controller, it coordinates the message exchange among the next hop domains about the QoS needed, the domains respond accepting or denying the request. The source domain controller uses the information obtained from the other domains to select and assign an optimal path to the flow. The results obtained showed that the overhead of the solution was smaller than when all the network information was broadcasted between the domains.

The paper (GENDA, 2021) proposes a bandwidth reservation method to meet high flow request acceptance and fast response requirements by combining ML and linear programming, particularly for unpredictable bandwidth demands in which the usage

time is strictly indicated. The users make the bandwidth reservation by a booking portal site. The request information includes the requested timeslot, source and destination nodes, and required bandwidth. When a request arrives, it is instantaneously judged as accepted or rejected through ML, then the network resources for all the logical paths are reoptimized using linear programming to solve a multicommodity maximum flow problem. Simulation results indicated that the proposed method provides suboptimal acceptance ratio and instantaneous request judgment.

TeaVisor enables each cloud tenant to specify the bandwidth requirements for its end entities (YOO et al., 2022). So, bandwidth requirements are entered per entity pair (user's computing entities, such as virtual machines and containers). When a packet-in occurs, the TeaVisor performs routing between the pair, establishing virtual paths. The bandwidth reservation component fulfills the bandwidth isolation guarantee by creating virtual paths combining multiple physical paths and reserving bandwidth's minimum requirements specified on switches links in a best-effort manner. TeaVisor satisfies the bandwidth requirements of a virtual path via multiple physical paths and rate-limiting rules. Many experiments were performed, such as Bandwidth Isolation Guarantee, Throughput and overhead. As presented, the framework performs competitive with other models, but seems worse in terms of overhead and throughput.

The paper (AL-HARBI et al., 2021) introduces a flexible QoS management model called Efficient Resource Allocation. It is developed a Graphical User Interface for users to facilitate the process of inserting QoS policies, creation of client's classes and customization settings of group applications and services for them. Then, the admission layer decides either to allow or reject additional reallocation or release of resources, grouping similar classes of flows into a queue. There is also a resources sharing mechanism between these queues, but it is not clear how it is set. The proposed model has been tested on different traffic (VoIP, HTTP, FTP, TFTP, and Video Streaming), and compared to FIFO (standard model), MAM (an example of IntServ model), and CBWFQ (an example of DiffServ model). The obtained results showed that, the proposed model was able to provide the best VoIP and video streaming quality, acceptance of Hypertext Transfer Protocol (HTTP) transaction delay, TCP connection delay, the lowest FTP and TFTP load delay.

The work by (TORRES et al., 2020) aims to propose a framework called Bandwidth Allocation Model through Software Defined Networking to address heterogeneous user and application requirements for dynamically allocating resources in MPLS SDN networks. The framework implements a Bandwidth Allocation Method (BAM) model by dividing the bandwidth into three traffic classes. Therefore, stakeholders wishing to transmit data through the network submit requests specifying a bandwidth amount and class. The BAM model then allocates resources for these flows along the

MPLS-labeled paths. However, this approach does not cater to individual flow requirements, as it categorizes them into broader and more abstract classes. Additionally, the model was designed to operate within a single domain, limiting its applicability in large-scale networks.

3.2 TRAFFIC CLASSIFICATION IN SDN

The work (RAIKAR et al., 2020) evaluates Support Vector Machine, GaussianNB, and Nearest Centroid models to identify applications such as HTTP, email, streaming, and VLC in SDN networks. The POX controller is built with a traffic classification architecture, where the data acquisition layer allows the collection of network topology information. The feature extraction layer consists of training models and AI algorithms for network traffic classification. The extracted traffic classification information is made available to the controller for decision-making. The decision and control layer receives the network state for decision-making and device control. To evaluate the ML models, real traffic was processed into features using "netmate", and labels were attached to identify this captured data. In the end, GaussianNB got the best results for precision 96.79%.

The work presented in (FAUZI; CHIN; ZAKI, 2024), explores the implementation of feature selection using filter and wrapper methods to tackle imbalanced datasets and enhance network management performance. Using a real-time dataset containing 930 features from the University of Cambridge, the Symmetrical Uncertainty to select the most important features and eliminate redundancy, and C4.5 algorithm achieve a classification accuracy of 97.69% and reduced the dataset size by 50%. The results showed an improvement of 15% in performance, comparing to traditional methods.

The paper (WANG et al., 2017) defines a new set of QoS classes and presents a modified K-Singular Value Decomposition (K-SVD) method for multimedia identification. Internet multimedia traffic was captured in a campus network. The datasets include 26 application types categories of multimedia traffic, which are divided into seven more broader QoS classes. The raw data is processed while the following characteristics are obtained, including protocol, port, payload, packet/flow size, flow duration, arrival time, IP address, etc. After analyzing their statistical characteristics, the datasets are processed using a scheme called bag-QoS-words. For training the methods, only 50 traffic flows of each class are selected. The classification results presented that, the proposed method surpass Naive Bayes, HMM, SVM and K-NN in general metrics.

In (EOM et al., 2021) the authors studied machine learning classification performance using ensemble learning for traffic classification in SDN. The classifiers were implemented at the network controller level, that was a RYU instance. They used the

same dataset as this current work (ISCX). The dataset was labeled in 7 traffic classes, but they didn't provide the data processing phase. Then, a couple of ensemble models were compared, such as Random Forest, LightGBM and XGBoost. Thus, the highest accuracy reached was 96.15% when using the LightGBM algorithm model.

3.3 BLOCKCHAIN AND QOS

The study (KARAKUS; GULER; ULUDAG, 2021) presents a novel blockchain-aided QoS-enabled inter-AS routing framework, QoSChain (QC), by blending merits of SDN and Blockchain technologies. Each blockchain node corresponds to an AS controller and runs its own instance. This way, the controllers update the blockchain with transactions, that are created from intern-paths along with their QoS values (maximum bandwidth and minimum delay) by blockchain nodes. A block contains transactions batched into a particular data structure. In the framework, when a QoS-based inter-AS service request comes from a user to an AS controller, the controller starts finding an E2E path considering the QoS parameters and priorities stated in the message using its blockchain ledger. When an E2E path is found satisfying the QoS requirements requested, the source-AS controller asks each AS controller over the E2E path to confirm whether they can provide the requested QoS values. Therefore, they evaluate and compare the performance of QC framework against two common QoS-based routing strategies in SDN networks: Hierarchical Routing Approach, and Distributed Routing Approach. The results show promising performance in terms of flow setup time and the percentage of requests serviced. The QC shows a comparable message overhead to the shortest-path based distributed approach based on BGP.

The paper (RAHMAN et al., 2021) presents an optimized energy-efficient and secure Blockchain-based software-defined IoT framework for smart networks, that ensures efficient cluster-head selection and secure network communication via the identification and isolation of switches. The work, proposes two different Blockchains, one for the control layer and one for the data layer, leveraging Ethereum as Blockchain platform. In the control layer, blockchain maintains the consistency of the flow rules of each cluster of controllers. On the other hand, in the data layer, all the switches dump their flow rules in the chain sequentially and verify if they are maintaining the same rule set. If any of the switches do not dump the same rules, the record is not updated, and the switch is isolated from the environment. This isolation helps to identify not only a fault in the switch, but also to contain adversaries if the switch is compromised. Simulations were conducted using Mininet with 100 IoT nodes transmitting packets ranging from 128 to 1024 bytes. Ultimately, the adjusted Ethereum solution was compared against the Blockchain Fundamental baseline algorithm, revealing that the proposal outperforms in terms of energy consumption and average throughput.

In (KHAN et al., 2021), was proposed a framework of blockchain-distributed technology-enabled QoS (QoS-ledger) computation in healthcare applications, in a permissionless public peer-to-peer network using smart contracts. The main goal was to allow analysis of real-time medical services and data processing. In this approach, fog nodes in the neighborhood collect medical and QoE data from patient devices. These nodes, store the proceed information in a distributed ledger. According to the authors, the proposed solution performs better in comparison with state-of-the-art methods.

The Trust aware E2E QoS Routing (TRAQR) framework is proposed in (PODILI; KATAOKA, 2021), for provisioning E2E QoS with trust and verifying the QoS compliance in multi-domain SDN. Every domain exchange information about QoS capabilities and recommendation trust scores. TRAQR uses the trust score as a metric to quantify trust of a domain. The domains in TRAQR select intermediate domains with higher trust scores to compute inter-domain QoS paths for E2E QoS requests. Upon completion of the E2E QoS delivery, the source domain securely obtains the logs that comprise QoS implementation results from all the domains along the E2E path and verifies their QoS compliance as well as accountability. The verification result of the QoS compliance is used to compute the trust scores of the intermediate domains. The updated trust scores are further used for the next round of selecting trusted inter-domain paths for routing E2E QoS traffic. Therefore, when an E2E QoS request arrives in a domain, the TRAQR manages the revisioning of E2E QoS along trusted intermediate domains, and verifying QoS compliance of intermediate domains. The results show competitive values between the TRAQR and multi-domain routing baseline algorithm, when there are multiple untrusted domains to find a path with an acceptable trust score.

3.4 CONSIDERATIONS ABOUT THE RELATED WORK

The SDN architecture has brought new paradigms for developing frameworks and mechanisms to implement QoS for flows. In this regard, the SDN controller plays a crucial role, as it has a global view of the network and enables treating each flow with specific rules and actions. Consequently, previous works have focused on leveraging SDN controller resources with the OpenFlow protocol to develop QoS implementation solutions or to support QoS.

Most of the related works that implement QoS frameworks are incomplete. Since QoS is an end-to-end concept defined by the service providers' ability to meet the requirements of service users, most works do not consider the requirements of data flows. Even when they do, they often fail to propose a model for end-to-end resource reservation. In general, these works monitor the network and accept flows in a best-effort paradigm.

In terms of traffic classification in SDN, the related works focused on comparative analyses. Generally, the machine learning methods themselves do not change, rather, what changes is the way databases are processed and features are extracted, which can yield different classification results. However, most works do not integrate the classification methods with mechanism of resource allocation and inter-domain QoS.

On the other hand, studies on blockchain implementation in SDN scenarios show great promise. These works aim to develop auditability and control of QoS by leveraging the distributed and secure characteristics of blockchain technology. However, these studies focus on methodologies for providing simple and less effective QoS. Therefore, there is much to explore regarding blockchain and multi-domain QoS.

Table 4 presents a descriptive comparison among the most recent QoS frameworks found in the literature. Each framework focuses on different aspects of QoS, such as routing or resource reservation, along with a traffic classification model. However, it is noticeable that these frameworks do not address multi-domain QoS provisioning, as they do not consider end-to-end resource reservation and don't implement an efficient traffic classification method. Furthermore, the comparative tests conducted do not evaluate different frameworks, which is an aspect to be addressed in future work.

Table 4 – Comparison of related works of QoS frameworks.

Reference	Objective	Method	Pros	Cons
(AZIZ et al., 2023)	Allocate network resources dynamically and predict traffic	Predict traffic usage and controls the bandwidth consumption per class (best-effort)	Can predict traffic with 97% accuracy in the experiments	Uses DPI as traffic classification. No bandwidth reservation.
(AL-JAWAD et al., 2021)	Multi-media-based routing method	Select the most suitable routing algorithm based on the network state and the application	Outperforms the algorithms when running alone	Uses port based traffic classification. No bandwidth reservation.
(THAZIN; NWE; ISHIBASHI, 2019)	End-to-end dynamic bandwidth allocation	User requests amount of QoS. A route will be defined based on the delay specs. A queue is selected based on the bandwidth and priority.	Results showed better throughput than others compared. Detects bottlenecks and reroute highest priority flows to alternative routes. Better throughput than other two methods compared.	No classification. User informs QoS. Three priority queues, no bandwidth reservation.
(OLIVEIRA et al., 2021)	QoS provisioning mechanism	Host request QoS by RSVP. The controller checks if it can forward the flow, then it sends by a queue. Otherwise, replies a forecast.	Results compete with other methods.	It is a very generic model. It doesn't determine QoS classes, routes or reservation model. The results can be controversy.
(GHALWASH; HUANG, 2018)	Framework for QoS route determination using OpenDayLight.	Calculates port utilization and reroute traffic in case of high delay and congestion.	Results showed QoS reduces the overall average delay by 57% in the evaluation, comparing with a L2 switch only scenario.	Very weak in terms of QoS provisioning. No traffic classification and no bandwidth reservation.
(JOSHI; KATAOKA, 2019)	Privacy preserving E2E QoS mechanism in multi-domain SDN	Multi-domains share routing QoS information. A domain receives a QoS request, then asks to next hop domains their QoS routing information. The edge domain analyses the information and the feasibility to forward the flow.	The method implies less overhead than others compared.	Interfere on the autonomy of the domains, because the edge domain decides for all. No bandwidth reservation or classification method.
(GENDA, 2021)	Framework for bandwidth reservation to provide high flow request acceptance and fast response using ML and linear programming.	Users request QoS by a web portal. A machine learning admits or not the flow. The linear programming defines a route for it.	ML classifies a network state and a request into accept or not. Linear programming minimizes the maximum link utilization.	Results suboptimal (inconclusive).
(YOO et al., 2022)	Framework for cloud tenants bandwidth allocation.	The pair of entities inform the bandwidth requirements. The framework will combine physical and virtual paths to fulfill the requests. Meter rules are used to limit traffic inside the paths. Provides a GUI for users to inform their group class and QoS requirements. The admission layer decides to accept or not the flow. Similar groups are put in the same queue.	The experiments performed bandwidth isolation, throughput and overhead testes. The results obtained are competitive with other models. But worst in some scenarios.	Different approach. Doesn't take flow requirements in account. No classification method.
(AL-HARBI et al., 2021)	Framework for efficient resource allocation.		The proposed model was able to provide the best VoIP and video streaming quality	No classification. No resource reservation. No multi-domain solution.

4 PROPOSAL

This chapter proposes the architecture for multi-domain QoS provisioning as a new E2E service on the Internet, as a second version of FLOWPRI-SDN framework. To recapitulate, currently, the Internet basically has two big divisions, the cloud service providers and the ISPs tier traditional architecture. Nowadays, most applications are hosted as cloud services. The ISPs sell connections to clients who are likely to access data from many cloud services per day. Within the cloud, application traffic might receive the QoS it needs to work. However, the ISPs still are based on legacy infrastructure and do not implement E2E QoS and, when they do, they are not transparent to each other about the QoS provided, lacking of trust between administrative domains.

Every application that needs the Internet to operate is designed to work under certain specific minimum requirements. The main problem is that traffic requirements are not advertised. This makes it harder to implement tailored solutions per flow, as it is not clear what is crossing the infrastructure of a domain. However, the cloud service that hosts application, might know this information but does not advertise it also. In FLOWPRI-SDN, it is considered that QoS flows are flows with a bandwidth, delay, jitter, and packet loss restriction, i.e., multimedia applications, games, and so on. The second version of FLOWPRI-SDN is proposed with new modules to face these challenges.

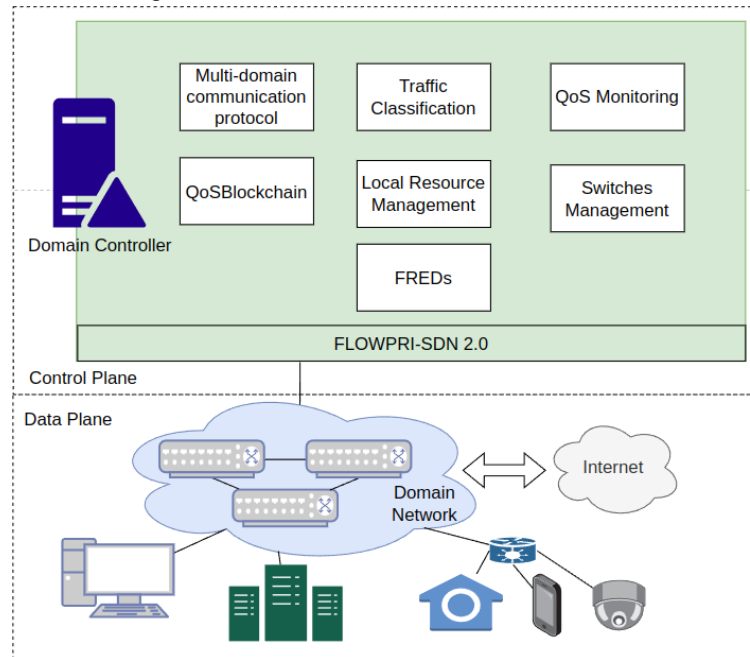
4.1 FLOWPRI-SDN 2.0

The FLOWPRI-SDN is a distributed SDN controller based on the RYU version 3.14 written in Python. It manages groups of switches of a specific network domain (like an ISP infrastructure) and has mechanisms to provide collaborative inter-domain QoS provisioning.

This new version of the framework is called FLOWPRI-SDN 2.0. In this version, it is intended to be able to address the issues of QoS provisioning, such as: QoS accounting for inter-domain trusting, traffic flow mapping into QoS requirements and inter-domain cooperation to provide QoS as a service to clients. The FLOWPRI-SDN 2.0 architecture leverages the local resource management (G-BAM) and the switch management models of the first version (described in Section 2.7) and is presented in Figure 21. The new proposed modules are described below:

- Identify flow's requirements: It implements Random Forest as the ML model and a different flow processing strategy to perform traffic classification.

Figure 21 – FLOWPRI-SDN 2.0 architecture.



Source: The Author, 2024.

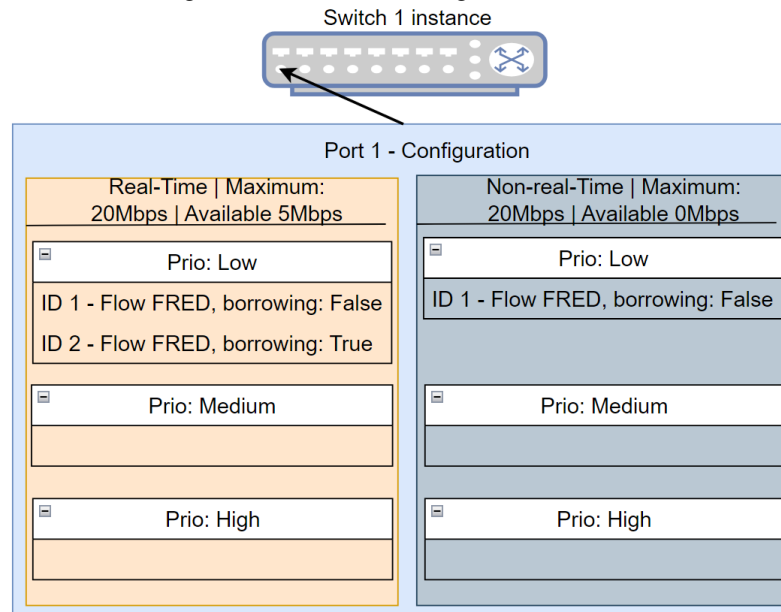
- Multi-domain E2E QoS: It uses the FRED and a new communication protocol to perform E2E QoS.
- Traffic monitoring: New strategy for traffic and QoS monitoring.
- Trust and QoS accounting: It uses the hyperledger sawtooth blockchain, to store QoS states for flows.

4.1.1 Switches Management

This module is inherited from the FLOWPRI-SDN first version, described in Section 2.7, and is responsible for configuring the domain switches. When the FLOWPRI-SDN controller is turned on, first, the domain switches must announce themselves; this is a default behavior for OpenFlow compatible switches. In the proposed framework, this triggers the switch features event and FLOWPRI-SDN stores an instance of that switch as an object, as Figure 22 shows. At this point, the FLOWPRI-SDN must be manually configured because topology discovery cannot identify the intended bandwidth that each port must have.

FLOWPRI-SDN reserves port 9999 to receive manual configurations. It accepts Java Script Object Notation (JSON) format configurations for switches, routes, hosts and OpenFlow rules. Listings 4.1, 4.2, 4.3 and 4.4, show the JSON format expected for each manual configuration. Therefore, after manually configuring the switches and their ports, the FLOWPRI-SDN is able to configure the queues of each port. In fact, the RYU

Figure 22 – Switch management module.



Source: The Author, 2025.

Listing 4.1 – Switch config JSON.

```

1 {"addSwitches": [
2   {"swich_name": "name",
3     "ports": [{"port_name": "port_name",
4                 "total_bandwidth": "bandwidth",
5                 "nextHopName": "nextHopName"}]}
6 ]}
7 }
```

Listing 4.2 – Route configuration.

```

1 {"addRoute": [
2   {"ip_ver": "version",
3     "src_prefix": "prefix",
4     "dst_prefix": "prefix",
5     "src_port": "port",
6     "dst_port": "port",
7     "proto": "proto"}]}
8 }
```

controller, that is the base of FLOWPRI-SDN, does not support queue configuration, however Linux based switches, like OVS, supports it, so it can be configured using their own queue tool (such as ovs-vsctl and tc-qdisc).

In each switch, the FLOWPRI-SDN divides the link bandwidth into four data traffic classes using HTB queues: control (7%), best-effort (25%), real-time (33%) and

Listing 4.3 – NAT config JSON.

```

1 {"addNAT": [
2   {"ip_private": "ip_translated"}]
3 }

```

Listing 4.4 – Host config JSON.

```

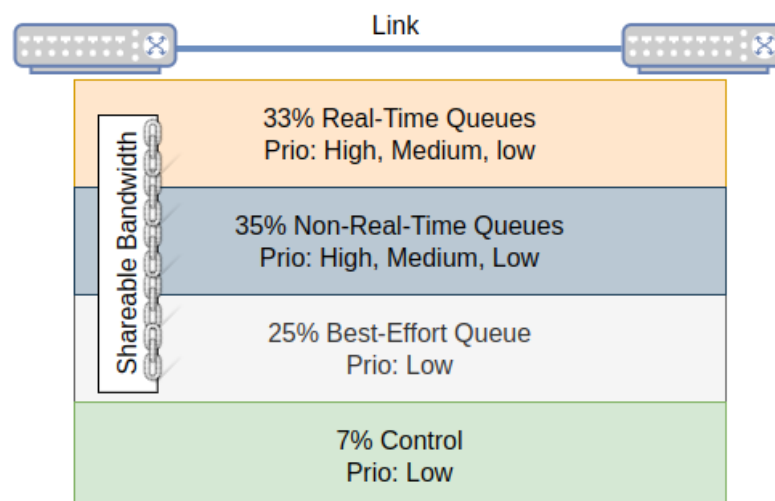
1 {"addHost": [
2   {"ip": "ip",
3     "ip_ver": "ip_ver",
4     "mac": "mac"}]
5 }

```

non-real-time (35%), as can be seen in Figure 23. Each class is configured with three priority queues (leafs) to allow different treatment for priority flows in terms of forwarding precedence and bandwidth shares.

The FLOWPRI-SDN combines queues with OpenFlow meter rules to manage bandwidth between those traffic classes. The G-BAM mechanism controls the bandwidth sharing for QoS flows inside the borrowings between QoS classes, while the HTB mechanism leverages the bandwidth that is not used from other queues to the best-effort queue in a non-reserved way.

Figure 23 – Switch management module.



Source: The Author, 2025.

4.1.2 FRED

In the FLOWPRI-SDN, the QoS requirements for a flow are stored in a document called FRED. It is also one of the main components for multi-domain QoS collab-

oration. A FRED is an individual document that carries the information about a flow, its QoS requirements and the route information. This document is generated whenever a new flow is identified using the traffic classification procedure and is forwarded to all the network domains where flow packets will transit.

The FRED structure is shown in Listing 4.5. It must carry the traffic flow 5-tuple information, the priority, traffic class, application type, the QoS forecasted (bandwidth, delay, and jitter), the classification label, the blockchain name, the source blockchain IP address range, the destination blockchain IP address range, the genesis node IP address, the pair list (node name, public key, IP address, and port to the QoSBlockchain).

The code field determines whether the network domain rejected the FRED (code=-1) or accepted it (code=1). The 5-tuple identification data identify the flow by its FRED. In the QoS fields are the flow requirements that are used by the G-BAM module for resource allocation. Lastly, the QoSBlockchain fields are important to form the blockchain network and connect the domain peers. Also, the blockchain needs the public key of each domain peer to perform the consensus protocol.

4.1.2.1 Multi-domain communication: Sending FREDS

As described in Section 2.2.1, there are some interesting ICMP pairs of messages for inter-domain communication, which are the messages type 139 and 140 for IPv6, and also 15 and 16 for IPv4 (proposed in FLOWPRI-SDN 1.0) (JÚNIOR; FIORESE, 2023). The FLOWPRI-SDN reserves the first messages of each pair for FRED announcement, whereas, it reserves the bigger values of the pair for informing the FRED's rejection.

The FREDS are announced by being encapsulated in the data field of an ICMP (15 or 139). Then the packet is sent with the same source and destination IP as the FRED has. If any address translation has to be done, it is done before sending the FRED. Then, the FLOWPRI-SDNs platforms on the route of the flow, will handle the ICMP in the packet-in module, as it is described next in Section 4.1.4.

A FRED rejection is motivated by the inability of some network domain to provide the resources announced in the FRED. In this case, an ICMP (16 or 140) is sent with the packet source as the destination of the FRED, whereas its destination is the source of the FRED. This allows previous network domains to stop providing QoS and handle the flow as a best-effort. The procedure responsible for dealing with FRED rejections is also described in Section 4.1.4.

Listing 4.5 – FRED structure data.

```

1 {"FRED": {
2   "code": 1,
3   "ipver": "6",
4   "proto": "TCP",
5   "ip_src": "2001:DB8::130F:0:0:140B",
6   "mac_src": "02:42:ce:63:af:f5",
7   "mac_dst": "02:42:41:df:3d:fa",
8   "ip_dst": "2001:12ff:0:4::22",
9   "src_port": "2345",
10  "dst_port": "80",
11  "priority": "1",
12  "class": "real-time",
13  "bandwidth": "2Mbps",
14  "loss": "0",
15  "delay": "30",
16  "jitter": "1",
17  "label": "ae-2Mbps",
18  "blockchain_name": "AS1000-AS2000",
19  "AS_src_ip_range": ["IP1-IP2"],
20  "AS_dst_ip_range": ["IP1-IP2"],
21  "ip_genesis_node": "2001:DB8::130F:0:0:140B",
22  "peers_list": [{"name_node": "name", "pub_key": "key", "
    ip_port_bc": "ip:port"}],
23  "route_list": [{"order": 1, "name_node": "name", "pub_key": "key",
    "hops": 5}]
24 }}

```

4.1.3 Local Resource Management

This module is also inherited from FLOWPRI-SDN first version, described in Section 2.7. It is activated whenever a new OpenFlow rule has to be created or removed. In the FLOWPRI-SDN 2.0, the difference is that the timeouts of the rules are adjusted, and sometimes the rules carry cookies, to manage when perform traffic monitoring (cookie = 1 means that the flow rule is being used for monitoring).

An OpenFlow rule is always associated with a switch port. It defines match fields that will be compared to each packet header, and actions for that packet. The actions can be: limit the forwarding rate (meter rules), change a header field, change a cookie (small integer attached to the OpenFlow rule) and forward the packet to a queue of an output port. In FLOWPRI-SDN, there are two types of OpenFlow rules: QoS rules and best-effort rules.

A QoS rule is composed of a unique OpenFlow meter rule that defines the maximum flow throughput. The meter rule is always attached to a forwarding rule. In

this way, if the packets matched in the forwarding rule exceeded the configured rate, it starts dropping packets to stabilize throughput.

The QoS rules are set with two timeouts, to control and renew the flow rules from time to time. The idle timeout removes the flow rules when no packets arrive in the switch after 2 seconds. The other timeout is the hard timeout that forces the flow rule to automatically remove itself after 5 seconds. Then, when the flow rule is auto removed from the switch table, the FLOWPRI-SDN receives a copy of the rule, to update its rule database removing and freeing the bandwidth that was used by it in the correspondent switch instance.

Best-effort rules are simpler rules. These rules do not have a meter rule attached, and they perform a simple action: put the packet in the best-effort queue of an output port. Also, they are set with the same values for timeouts, to allow the flow to be re-classified in case the flow has changed its QoS requirements after the last classification.

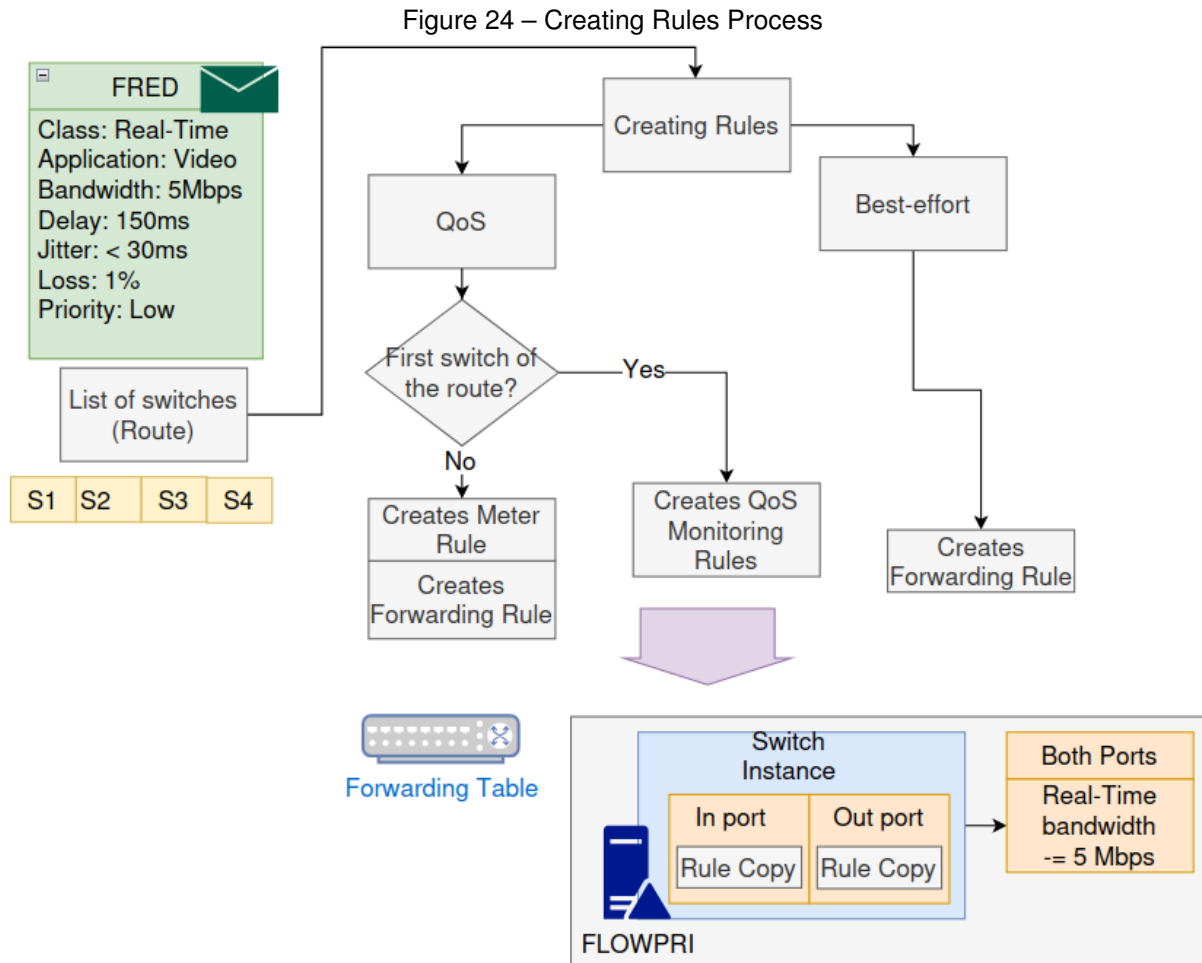
Figure 24 helps to understand the steps taken to create new rules with FLOWPRI-SDN. When OpenFlow rules are created in a forwarding table of a switch by the G-BAM module, a copy of that rule is stored in the `in_port` and `out_port` instances of that switch, in the list for the queue for that flow. As described before, there are three priority queues for each QoS class, one best-effort and one for control rules. QoS rules reduce the available bandwidth of the QoS class in the same amount that was configured for the meter rule attached to the rule flow. This done for the `in_port` and `out_ports` from which the flow will cross in the switch. However, the best-effort rules do not reduce the bandwidth available because these types of flows are not shaped, then the HTB queue limits by itself.

The rules can be removed in active or passive modes. In the active mode, FLOWPRI-SDN sends a message to the switch, ordering to remove the rule that matches certain fields. In the passive mode, the switch removes the OpenFlow rules that have expired the timeout values. In both cases, a copy of the rule and the reason are encapsulated in a message that is sent to the controller.

The FLOWPRI-SDN handles this message by gathering the switches of the route for that corresponding flow, removing the rule from all the instances, freeing the bandwidth that has being consumed. If it was a QoS rule, it removes the meter rule associated and the flow's FRED too.

4.1.3.1 QoS Monitoring Rules

The FLOWPRI-SDN 2.0 implements a traffic monitoring strategy to keep accounting of the QoS provisioned and the QoS expected. These rules are created when



Source: The Author, 2025.

a QoS FRED is accepted. The model that leverages these rules will be described in Section 4.4. The QoS monitoring rules are the normal QoS rules described before, also created by the G-BAM module, but that are created only in the first switch of the route of the flow. The other switches are configured with normal QoS rules. The reason is that, only one switch must send packets of a QoS flow to the FLOWPRI-SDN to be monitored, and it must be done periodically.

Normal QoS rules expire by idle timeout and hard timeout, then the flow becomes best-effort for a while until the next classification. During this period, it does not worth to monitor the flow, because the switches are handling it as best-effort. Therefore, the QoS flow must be monitored during the QoS rule activity.

The FLOWPRI-SDN differentiates one normal QoS rule from a monitoring QoS rule using cookies. A cookie is an integer field of the OpenFlow rules, that can be set by the controller, and it is retrieved when the flow expires.

To perform the monitoring while the QoS rules are active, the first of the switch route has a different QoS rule, QoS monitoring rule. This rule expires at each 2 seconds

by hard timeout, then the copy of the rule is sent to the FLOWPRI-SDN. This rule has two phases, monitoring phase and normal behavior phase.

In the monitoring phase, the cookie is set to 1. The packets are marked, and a double action is configured. The packets are normally forwarded to the output port and queue in the switch, and a copy of the packet is sent to the FLOWPRI-SDN. After receiving 20 packets, the FLOWPRI-SDN updates the rule with cookie 0, and hard timeout of 2 seconds. Thus, the rule will send flows normally until it expires, and the rule will be created with cookie set to 1.

This strategy allows the FLOWPRI-SDN to monitor the traffic two times before it expires in the other switches, collecting values for QoS metrics of 20 packets. The metric values are exchanged with the destination edge FLOWPRI-SDN using ICMP (15 or 139) for the QoS monitoring model, that is explained in section 4.4. Then, when any other switch of the route of the flow has the flow rule expired, the QoS rules are removed to perform a new traffic classification.

4.1.3.2 G-BAM

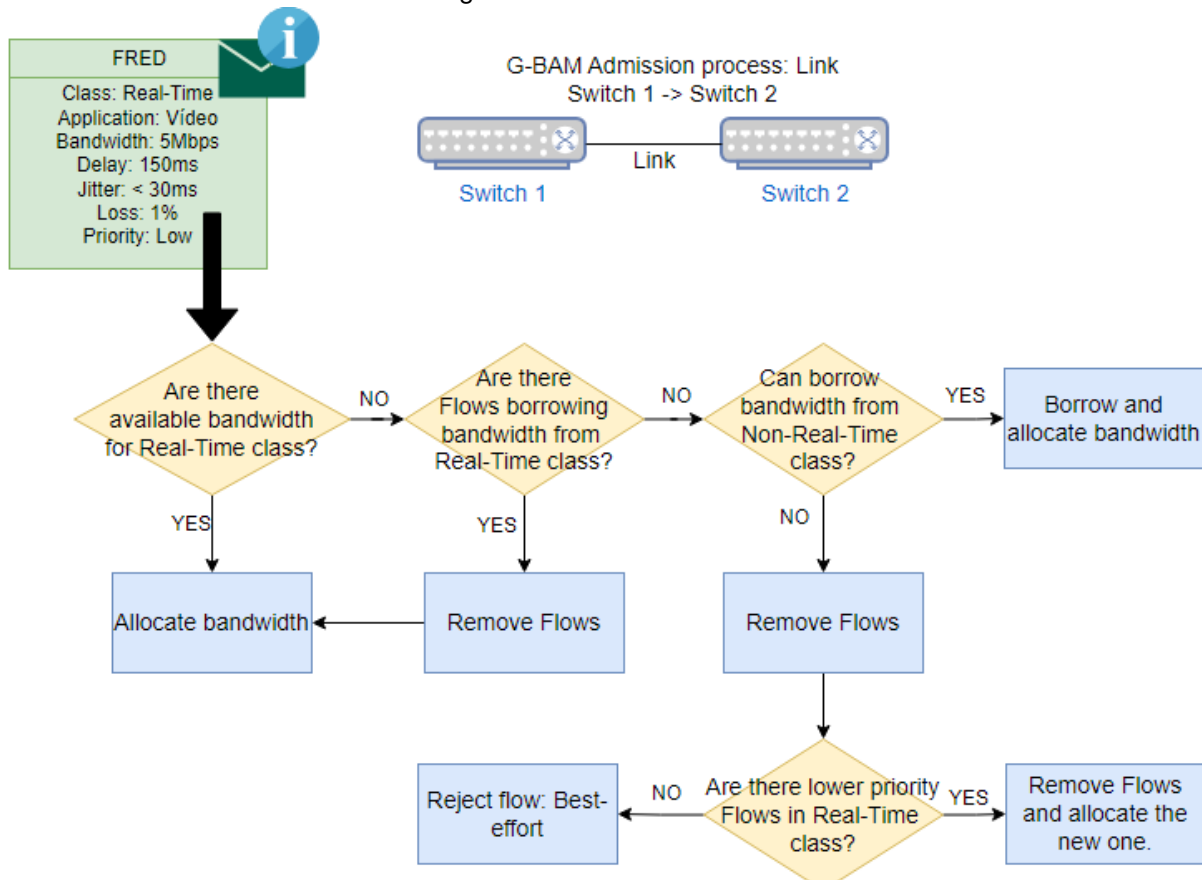
The G-BAM module is responsible for the resource allocation in the switches, it chooses the right queue to forward the packets of a flow and the logic for bandwidth sharing for QoS flows. This module is also inherited from the FLOWPRI-SDN first version. In the process, the packet-in event obtains a FRED from the traffic classification method or from an ICMP packet. The FLOWPRI-SDN finds a list of switches that compose the flow's route. Then, G-BAM checks the QoS bandwidth available in the output and input ports of all the switches of the route based on the QoS fields of the FRED and tries to reserve bandwidth for the flow.

There are two QoS classes with different amounts of bandwidth reserved: real-time and non-real-time. Figure 25 helps to remind the steps to allocate bandwidth with G-BAM. First, it gets the flow's class described in the FRED and checks the bandwidth in the queue for that class to create the QoS rules. The phase of creating rules for a flow, follows the steps described in Section 4.1.3.

In case it does not have enough bandwidth, it removes the flow rules that are borrowing bandwidth in the class. Then, in case it fails for gathering bandwidth, the G-BAM tries to borrow bandwidth from the other class. However, if borrowing is not possible, the G-BAM tries to remove less priority QoS flows from the original class queue.

In case all previous actions fail to allocate bandwidth, the FRED is rejected. Therefore, the FRED's code is changed to -1 and is encapsulated inside an ICMP packet. This packet is sent to the flow source host with the host destination as its

Figure 25 – G-BAM flowchart.



Source: Adapted from (JÚNIOR; FIORESE, 2023).

source, making all the previous network domains on the route to reject the FRED as well.

4.1.3.3 Backbone G-BAM

The backbone network domains are not able to provide per flow QoS allocation. In fact, the traffic that is shaped once, does not need to be shaped twice. This way, the source edge domains' FLOWPRI-SDN shape the QoS flows using meter rules and allocates bandwidth in QoS queues. So that, the next domains of the route only have to control the limit of bandwidth consumed by each QoS class, once the flows are limited. The amount of OpenFlow rules in each switch would break the memory limits. In these cases, it is proposed a best-effort inside the QoS queues to mitigate the bandwidth reservation.

This strategy proposes that the backbone domains (a domain that is not the source and not the destination for the flow) groups flows based on the FRED document received. Then, the sum of bandwidth of the FREDs of a class represents the amount of bandwidth that has been consumed by them. It allows that one big OpenFlow rule, can group many flows in a QoS without the risk of decreasing the QoS level registered

in the FRED.

Then, every time a new FRED arrives, the list of FREDs are checked to remove the flows that are going for more than 5 seconds (hard timeout). In this case, it mitigates the impossibility of implementing idle timeouts, per flow bandwidth allocation in backbone networks, also keeping a small forwarding table in the switches.

As the QoS flows are limited by meter rules in the edge network domains, then, all the flows in QoS rules are shaped. It allows controlling the bandwidth utilization for each class by using the active FREDs that are stored in the FLOWPRI-SDN.

The backbone domain rejects new FREDs when the sum of the bandwidth of the already stored FREDs are equal the bandwidth reserved for the corresponding class. Then, every 5 seconds, the hard timeout is activated in the edge network domains, and a new FRED will arrive to update the FRED stored.

This allows the FLOWPRI-SDN to remove FREDs that are stored for more than 5 seconds, freeing (logically) the bandwidth available for the corresponding QoS class. This strategy also allows to backbone network domains, to group many flows by the FRED's QoS class into one best-effort OpenFlow rule, that enqueue in the QoS queue corresponding to the FRED.

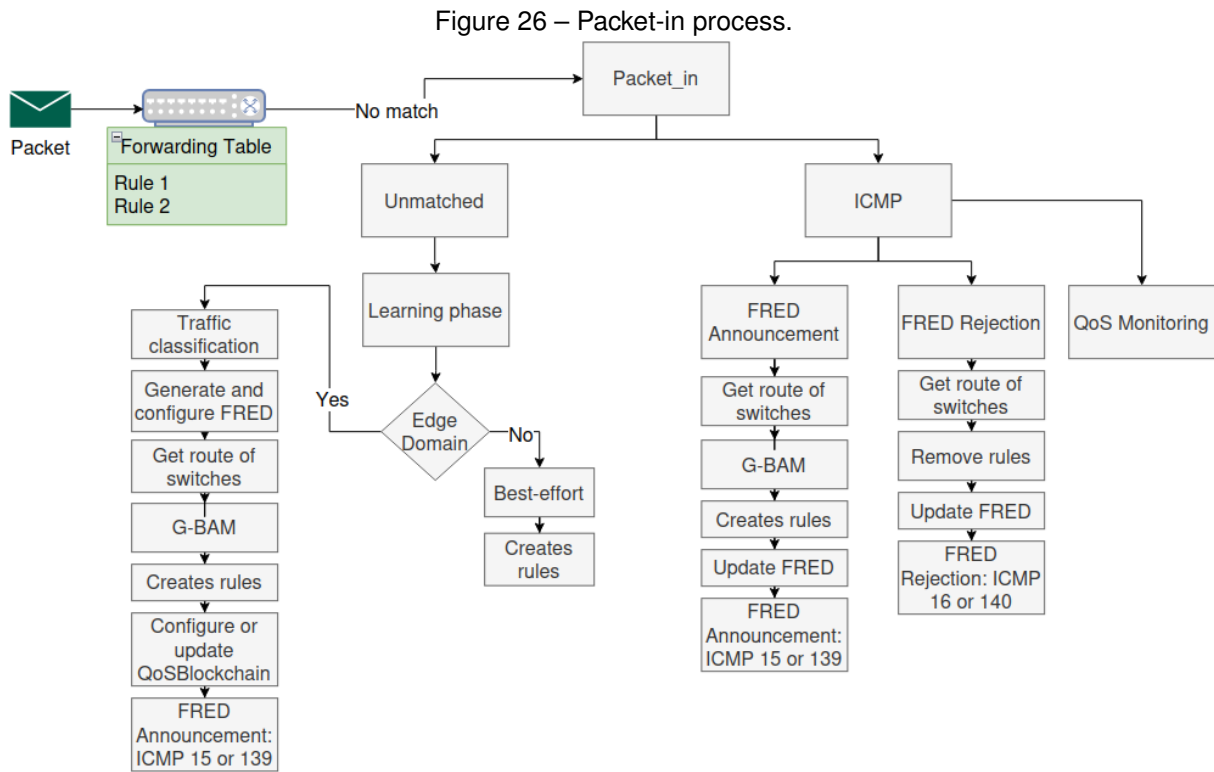
These backbone rules are created using the *Conjunctive Rules* (PFAFF; PETTIT; TOURRILHES, 2021). The conjunctive rules allows the OpenFlow switch to match multiple values to the same field and attach them to an normal forwarding rule. Each conjunctive rule has an unique ID (integer) and may be represented by multiple clauses.

The clauses may have multiple matching fields with one value each (like any other rule), but also, have a clause number and a number of clauses to combine. The clause number defines what which clauses must be combined. This way, a conjunctive rule combine multiple clauses with the same ID to form the list of values for the matching fields. The conjunctive rules are not well specified in the OpenFlow and RYU documentations, but they are implemented and available since OpenFlow 1.1.

4.1.4 Packet-in event handler

The packet-in handler is the core of the FLOWPRI-SDN 2.0 framework, where most of the work is done, and has basically the same face as in the 1.0 version. The FLOWPRI-SDN handles two types of packets: unmatched packets and ICMPs. The first switch of the route between two hosts will always send the packets to the FLOWPRI-SDN (packet-in event), if there is no OpenFlow rule preconfigured to match the header fields of the packet. It occurs when a packet from a new flow arrives at the switch and when a flow rule has expired (flow removed event), both types of packet are handled this way.

Figure 26 describes the steps taken to process a packet in the packet-in event. These steps are described in the following subsections.



Source: The Author, 2025.

4.1.4.1 Handling unmatched packets

Unmatched packets that arrive to the FLOWPRI-SDN are parsed in the packet-in event. The first thing is to learn the hosts' information, such as: MAC addresses, IP addresses, and in which port of the switch is the source host connected. Then, the route of switches for the flow is calculated (external API or manually configured).

At this point, the FLOWPRI-SDN must identify if the source of the packet is a host from its domain, then it is an edge domain. If the host is not from its domain, the classification is not performed, and the flow is handled as a best-effort one. Otherwise, it starts the traffic classification process to compute the FRED for that flow. The classification process will be explained in Section 4.2.

As soon as the flow is identified and the FRED is created, two scenarios can occur for the flow: it is classified as a best-effort or a QoS one. In the best-effort scenario, the OpenFlow rule is updated to keep it as a best-effort flow.

Otherwise, if the flow is classified as a QoS one, all the switches of the route run the G-BAM allocation method using the FRED. If all of them accept the FRED, the OpenFlow QoS rules are created in every switch of the route replacing the best-effort

rules previously configured. However, if the FRED is rejected, then it keeps the flow as a best-effort and sends an ICMP to inform the decision, toward the source of the flow (explained in Section 4.1.2.1).

In the case of FRED is accepted and one of the hosts (source or destination) is in its domain, then occurs the QoSBlockchain setup. This method configures the necessary information in the FRED for setting up the blockchain. As will be described in Section 4.3, each pair of edge domains, keep a QoSBlockchain to maintain the accounting of the QoS provisioning for their clients. A domain can be considered an edge domain if it is the network domain closest to the source host or the destination host of the flow.

4.1.4.2 Handling ICMPs

When an ICMP comes to the FLOWPRI-SDN by packet-in event, it can be of many types, like echo request or reply, for example. However, the types that FLOWPRI-SDN uses for QoS provisioning and inter-domain communication were described in Section 4.1.2.1

If a FRED announcing message arrives (ICMP 15 or 139), the FLOWPRI-SDN extract the FRED from the data field and runs the resource allocation method (G-BAM) to create the corresponding QoS rules. Then, the FLOWPRI-SDN adds its information in the `peer_list` (if this is an edge domain) and `route_list` of the FRED, before forwarding the ICMP to the next network domain.

In the case FLOWPRI-SDN gets an ICMP with a FRED rejection (ICMP 16 or 140), the FLOWPRI-SDN gets the switches of the route for that flow and removes the QoS rules, freeing the reserved bandwidth. The controller creates best-effort rules to replace the QoS ones, allowing the transit to be best-effort. Then, the ICMP with the FRED rejection is forwarded to the source of the flow, causing other domains to reject the FRED in the same way.

There are some cases when the ICMP (15 or 139) is encapsulating a JSON structure of QoS monitoring metrics and flow identification fields. In this case, the QoS monitoring model consumes this data to analyze the QoS provisioned for the flow and to create transactions in the QoSBlockchain.

4.2 TRAFFIC CLASSIFICATION

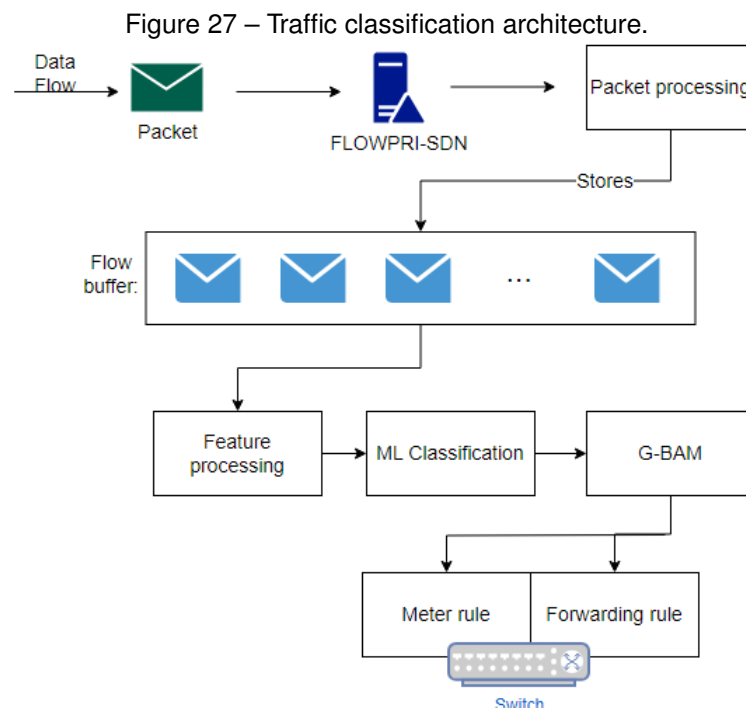
The traffic classification component provides the flow identification and QoS mapping to generate the flow's FRED. When an unknown traffic flow arrives at the switches managed by a FLOWPRI-SDN framework, it is sent to the controller via packet-in event. The traffic classification module takes 10 packets of the new flow to

conclude. Each flow has a buffer, that is updated with a new packet. If the buffer is completed, the traffic classification method classifies the flow and generates the FRED for it.

While the buffer is getting filled, the classification phase is not done and more packets from that flow must be collected. In this process, when the first packet is put in the buffer, the FLOWPRI-SDN 2.0 creates an OpenFlow rule to send the flow as a best-effort and also keeps sending a copy of the packet to itself (double output port rule). This rule is configured only on the first switch of the route. The other switches on the route will have a normal best-effort rule (no double output port rule).

In this way, the classification is performed while the flow is forwarded normally. The FLOWPRI-SDN first version had the hosts to be responsible for providing the flow requirement information. Nonetheless, some clients might not know their requirements for each application data flow and they can lie to receive more resources. However, this new module performs traffic classification without involving hosts for that.

Subsequently, after collecting 10 packets, the sample is processed to generate a set of feature values that serve as input for the Machine Learning (ML) classifier. After the classification finishes, it generates a FRED and G-BAM will process accepting or rejecting the flow. Figure 27 represents such a process.



Source: The Author, 2024.

Table 5 presents the set of features calculated for each flow sample. Among the computed information are the size of the largest packet in bytes, the sample duration, the sum of packet sizes in the blocks, Inter-arrival Time (IAT) between packets, payload

size of the packets, and other information related to TCP header fields. These metrics aim to represent applications (data flows) with similar QoS requirements and belonging to the same service class exhibiting similar statistical characteristics.

Table 5 – Set of features calculated for classifying flows.

proto	duration	pkt_sum_lenght
pkt_longest_lenght	pkt_shorter_lenght	pkts/seg
1quartil_pkt_lenght	3quartil_pkt_lenght	pkt_lenght_mean
pkt_lenght_median	pkt_lenght_std	pkts_lenght_menor_mean
pkts_lenght_maior_mean	payload_sum	payload_shortest
payload_longest	payload_mean	payload_std
payload_shorter_128	payload_128_1024	payload_longer_1024
1quartil_payload_lenght	3quartil_payload_lenght	IAT_shortest
IAT_longest	IAT_sum	IAT_mean
IAT_longest_medan	IAT_shortest_mean	IAT_std
1quartil_IAT	3quartil_IAT	tcp_push_flags
tcp_urg_flags	tcp_syn_flags	tcp_fin_flags
tcp_rst_flags	tcp_ack_flags	tcp_flags
tcp_windowsize_sum	tcp_windowsize_mean	tcp_windowsize_std
header_lenght_sum	header_lenght_mean	header_lenght_std

Source: (JR; PARPINELLI; FIORESE, 2023).

The class labels are the possible classification outputs of the traffic classification model. The class labels defined for this method were extracted from a paper published in (JR; PARPINELLI; FIORESE, 2023). In the referred paper, previous works such as the presented in Section 2.4 were extended to identify specific application QoS requirements, looking for it in their websites, then grouping the applications into classes of services. This way, for this work, the highest bandwidth for a class of service was chosen to be the class label in which this application traffic flows will be classified utilizing the ML model. Table 6 synthesizes the application classes and the bandwidth defined for each class. Therefore, with the class of service (delay and jitter) and the application type (bandwidth), the FRED of the flow can be generated.

4.3 QOSBLOCKCHAIN

A restriction for providing QoS end-to-end QoS is the trust between ASs and customers. For this reason, a blockchain solution is proposed to make the FLOWPRI-SDN 2.0 a trustful framework for multi-domain QoS implementation.

The QoSBlockchain is a Hyperledger Sawtooth application. This blockchain platform is an open-source allows peer-to-peer networks to maintain a permissioned and distributed database. The Sawtooth architecture is composed of the validator node that stores the blockchain transactions, and the clients that access the validator node

Table 6 – Application bandwidth and label defined.

Class of Service	Application	Bandwidth	Label
Real-Time Audio	VoipBuster, Skype, Hangouts	55-90kbps (codec) 100kbps (skype)	ar-100kbps
Static Audio	Spotify	1-2Mbps	ae-2Mbps
Real-Time Video	Skype, Hangouts	HD 1.5Mbps (skype) SD 300-500kbps (skype)	vr-1.5Mbps
Static Video	YouTube, Netflix	(Youtube) 1080p,30fps = 5 Mbps 720p,30fps = 2.5 Mbps 360p,30fps = 700kbps	ve-5Mbps
Chat	Facebook, Hangouts, ICQ, AIM, Skype, GmailChat	chats in general 24-128kbps	chat-128kbps
Data Download	Bittorrent, ftps, scp	Max (10Mbps)	down-max
Data Upload	ftps, sftp	Max (10Mbps)	up-max
E-mail	Gmail	10kbps	email-10kbps

Source: (JR; PARPINELLI; FIORESE, 2023).

to query new transactions. In this section, QoSBlockchain is proposed for storing traffic flows QoS metrics.

In the QoSBlockchain network, QoS provisioning is divided in two parts of interest, the clients and the network administrators. The ASs and Internet providers want to reduce costs, sell new services, improve the management of the network domain and retain customers. Therefore, network administrators need to know what resources each network flow requires and keep track of resource usage, so that they can provide QoS as a service. Whereas, customers want to access their applications without issues, pay a reasonable price, and understand the services they are receiving.

4.3.1 QoSBlockchain architecture

The end-hosts accessing network application might have traffic flow crossing multiple networks from one host in the source edge to the other in the destination edge. As described in the previous sections, both edges are managed by different ASs or network providers. Then, for each pair of end network providers, the FLOWPRI-SDNs must keep a QoS blockchain for tracing the communication between them, called QoSBlockchain. This mechanism is called by the QoSMonitoring module, to store QoS measurements (FRED and QoS values) provided to flows over time.

The QoSBlockchain is an application based on the Hyperledger Sawtooth platform. The blockchain architecture involves five processes that every node of the blockchain network must run (usually running with Docker):

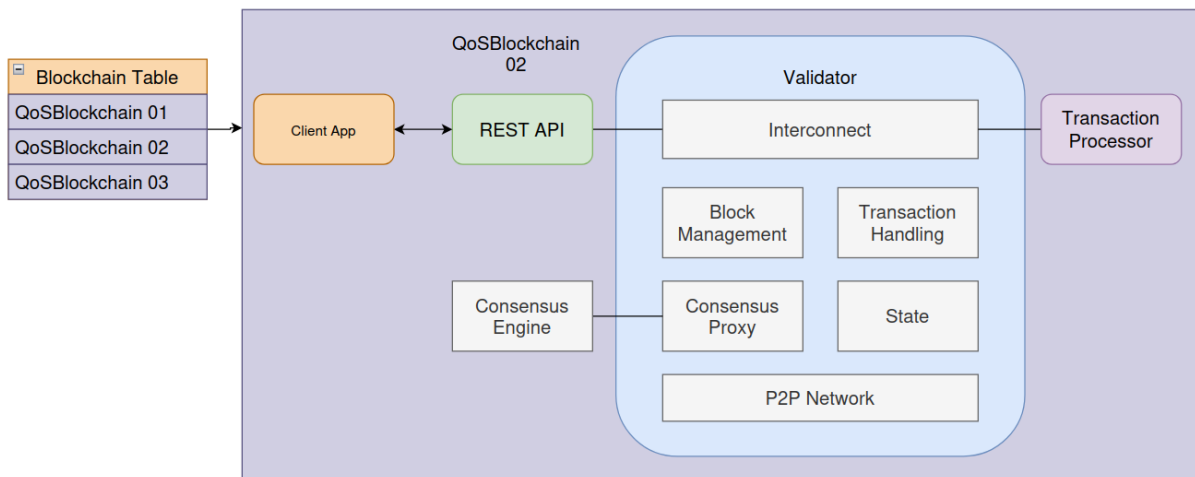
- Rest-API: It is the interface for communication between the validator node and

the client application.

- Transaction Processor: This is the business logic for the application of the blockchain. It includes the way of addressing blocks and how the transactions are stored.
- Validator: The core of the Hyperledger Sawtooth blockchain.
- Consensus mechanism: It validates the transactions.
- Settings: It is the database for the configurations of the blockchain.

The validator nodes (peers) are the pair of FLOWPRI-SDN 2.0 controllers that manages the ASs on these edges and one specific host of each domain. This host is a specific host that serves only for network management purposes (called management host). For the client hosts to interact with the QoSBlockchain, they need to forward the transactions to the management host. This division ensures that the blockchain network is separated 50% to customers and 50% to network administrators, and that one node alone cannot take over the database. Then, every node can maintain several QoSBlockchains, depending on the amount of pairs of end network domains that are in connection. Therefore, the management hosts and the FLOWPRI-SDNs, keep a blockchain table that maps the FREDs to the QoSBlockchain they have to send the transactions. Figure 28 displays the Sawtooth QoSBlockchain architecture.

Figure 28 – Sawtooth Node Architecture



The transaction processor is the main core of the QoSBlockchain. It specifies the payload of the messages and how the validator will process the transaction and update the global state of the blockchain. Moreover, the transaction processor defines the supported actions performed by the blockchain, such as consulting the transactions and how the transactions are stored in the blockchain.

Listing 4.6 – QoS block of data.

```

1 { "flow_name": "",
2   "qos_register": [] }

```

Any host that is a client of the pair of edge domains is considered a user of the QoSBlockchain and can perform three main actions to interact with the QoS registers of the application flows. First, hosts can query all transactions stored on the blockchain, so they can check how ASs deal with QoS in each route. In addition, they can check how many of its traffic flows the ASs served with the QoS specified. This information can motivate network improvements for both sides, customers and providers.

Second, hosts can query historical information on a traffic flow in the format of a FRED and QoS monitors. Thus, hosts can follow the QoS consistency of the service provided. Moreover, hosts that are the source or destination of a traffic flow can register a new state of QoS for current traffic flows.

Sawtooth stores data in a Merkle-Radix tree. Data are stored in leaf nodes, and each node is accessed using an addressing scheme that is composed of 35 bytes. However, the encoding of the address is completely up to the application (FOUNDATION, 2024). In QoSBlockchain, each leaf stores the QoS history of a traffic flow. The address is a hash from a combination of the 5-tuple identification fields plus the IP version, as well as the block names: IP addresses, IP version, protocol, and ports.

Every time a node sends a QoS register, the transaction processor runs the consensus mechanism to validate that. If it is valid, the transaction processor builds the address of the leaf, gets the node data stored, updates the values adding the new QoS metrics (expected and calculated) in the respective list, and stores it again, changing the global state of the Merkle-Radix tree. The Listings 4.6 and 4.7 show a block of data of QoSBlockchain.

This blockchain uses a private network design and leverages the Sawtooth permission control model. Sawtooth carries not only the ability to control validator's permissions but also transactor permissions. The local validator configuration limits who are allowed to submit batches and transactions directly to that validator. In the client API, the local permissioning mechanism allows only the source or destination of the traffic flow sending a new QoS transaction to the blockchain. Whereas, on the network domain controller's side, network-wide on-chain permissioning guarantees only validators known as peers to perform QoS updates in the blockchain.

The Sawtooth validator builds the network and performs peer connectivity, discovery, and message handling. A specific interface and port are used for incoming connections on the validator component to listen to new connections and handle peering,

Listing 4.7 – QoS register.

```

1 { "node": "2001:DB8::130F:0:0:140B",
2   "route_nodes": [],
3   "blockchain_nodes": [],
4   "service_label": "",
5   "application_label": "",
6   "req_bandwidth": 5000,
7   "req_delay": 150,
8   "req_jitter": 10,
9   "req_loss": 1:
10  "bandwidth": 5000,
11  "loss": 1,
12  "delay": 150,
13  "jitter": 10}

```

and exchange messages with each other nodes. Only static networks are compatible to QoSBlockchain.

The static network model requires the knowledge of all the nodes before setting the blockchain. The nodes are paired in bidirectional way, so that only one node need to know the other's IP address and public key each time. Then, the QoSBlockchain uses the PBFT consensus mechanism to validate transactions.

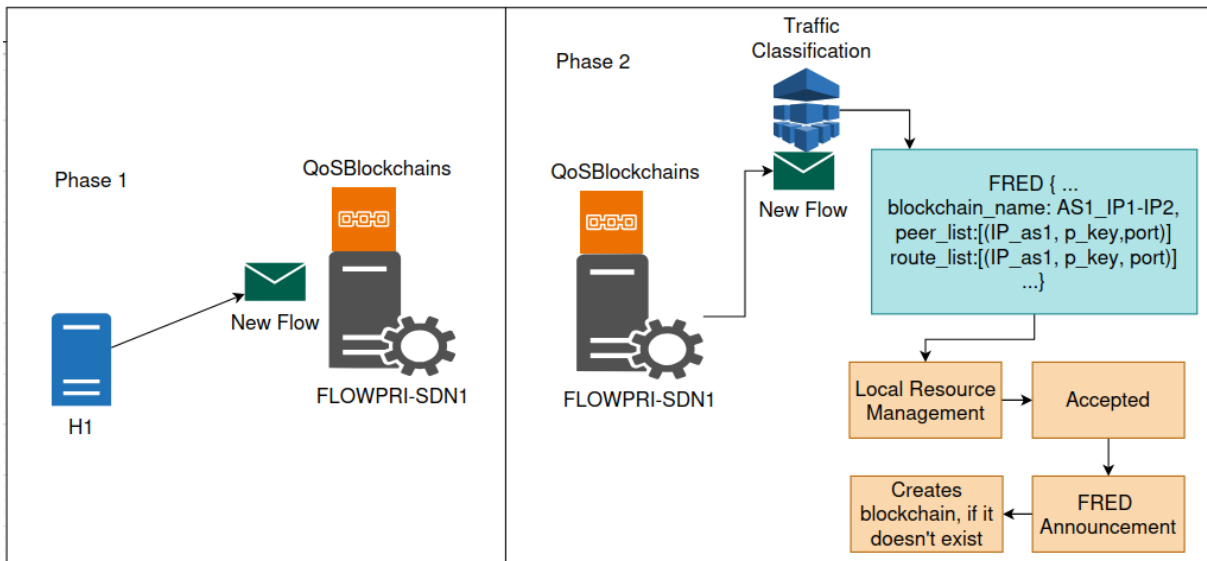
4.3.2 QoSBlockchain Setup

A QoSBlockchain is initiated when the network domain of the closest edge starts to transmit a new traffic flow that is classified and accepted as a QoS flow. First, it classifies the flow as a QoS one and verifies that there is no blockchain to register QoS between the two end domains in the blockchain table. In this case, FLOWPRI-SDN writes a FRED document, encapsulates it in an ICMP message (code 15 or 139) and sends it to the flow destination. These processes are represented in Figure 29.

At this stage, the FLOWPRI-SDN fill the FRED document with 5-tuple traffic flow information and the forecasted QoS. It will define the QoS goal that all administrator domains must configure. In blockchain fields, the genesis node is the management host on the source edge network domain. The source IP range for the blockchain is set, along with the blockchain name, that is a combination between administrator name with source and destination IPs of that FRED, that will form a unique name. It adds its identification name, IP and public key to the peer list and the route list. Thus, it starts the QoSBlockchain and waits for peer connections.

Only the edge network domains keep the blockchain. The other network domains on the route, just modify the FRED, writing their IP address and public key in the

Figure 29 – Blockchain setup phase 1 and 2.



Source: The Author, 2025.

route_list (in order). So, they are presenting themselves as part of the route and will respect the FRED, or they can reply denying the FRED. It means that it is not possible to provide QoS at the time.

When the FRED is accepted and configured by an edge domain, it is forwarded towards the FRED's destination host, until it reaches the other edge network domain. The inter-domain communication uses ICMPs to carry the FRED between network domains, and was presented in Section 4.1.2.1.

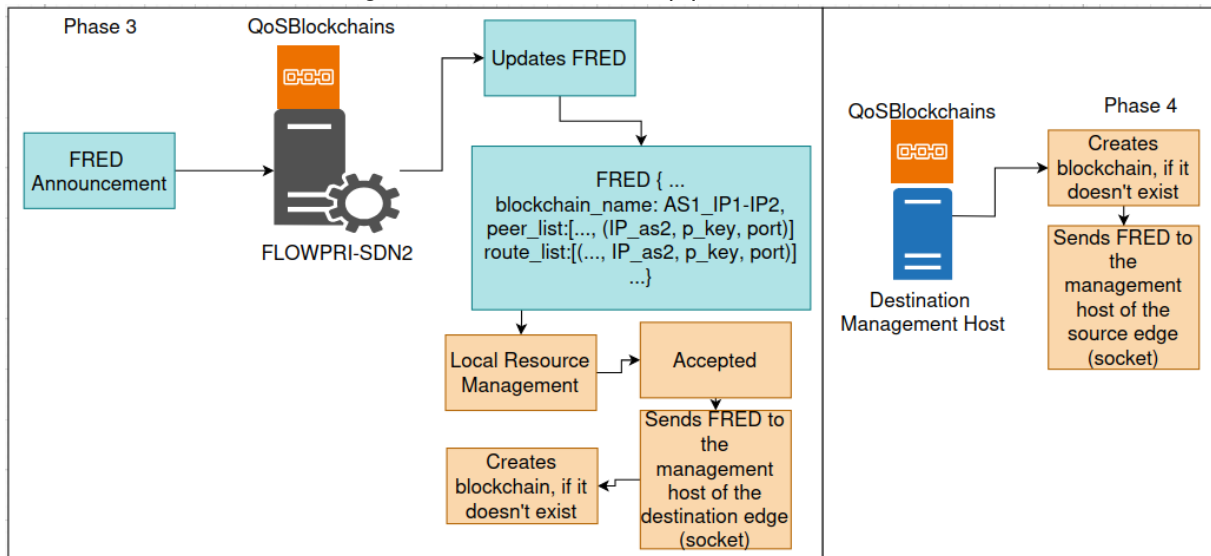
At some point, the ICMP carrying the FRED reaches the edge domain of the destination host of the FRED. In this case, the QoSBlockchain setup occurs normally in the FLOWPRI-SDN. Then, the FRED is sent to the specific management host using a socket. Figure 30 shows these processes.

Thus, this host writes its IP along with its public key in the peer_list and sends to the FRED server of the genesis host. To start the QoSBlockchain, it configures the blockchain with the peer_list received within the FRED. Then, this host sends the FRED using socket to the management host of the source edge domain, that is described in the FRED, as seen in Figure 31.

The source management host is always the genesis host of the QoSBlockchain. The genesis host configures the QoSBlockchain with the three peers IP, creates the genesis block, allowing the blockchain nodes to start registering the QoS for the traffic flows between the two network domains and creates the first transaction to register the QoS with FRED received. With the genesis host up, the QoSBlockchain for the pair of end network domains is configured and ready.

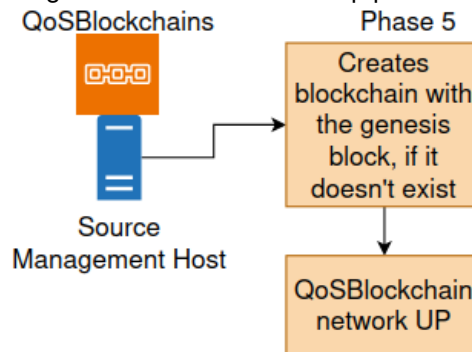
After the QoSBlockchain setup, hosts can ask the management host to send a

Figure 30 – Blockchain setup phases 3 and 4.



Source: The Author, 2025.

Figure 31 – Blockchain setup phase 5.



Source: The Author, 2025.

QoS transaction. The management host checks the IP address of the host and the IP address of the FRED. If they are the same, that host can send the transaction. Thus, the management host finds the right QoSBlockchain and stores the new block.

4.4 NETWORK MONITORING

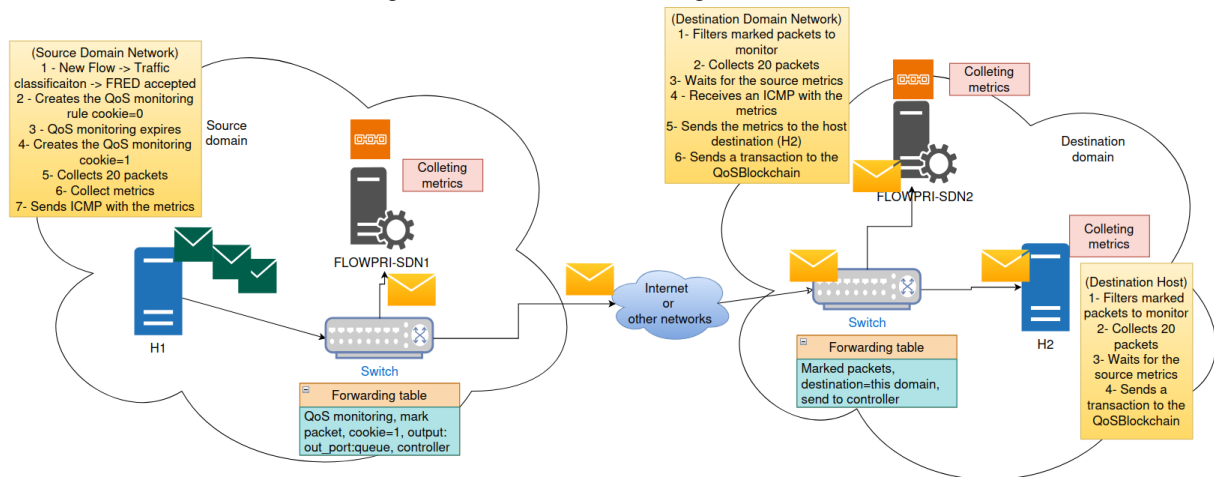
It is proposed a QoS monitoring method for end hosts and edge network domains to maintain a concise database of the QoS provisioned using the QoSBlockchain. The method is intended to mitigate the trust issues among the different ASs and to motivate multi-domain guaranteed QoS.

This approach, can help hosts and administrators to identify when QoS cannot be guaranteed and inconsistent QoS provisioning. Therefore, in scenarios with inconsistent QoS provisioning, network domains (AS) can stop providing QoS, saving computational efforts and being transparent to client hosts about the decision. Traffic monitoring is performed to keep a track on QoS levels for traffic flows. This increases

the confidence of operating QoS among ASs and reduces the risk of violating QoS assurances for sensitive applications.

Therefore, hosts and edge frameworks must monitor the QoS by collecting metrics (bandwidth, delay, jitter, and loss) of flows periodically and sending as a log of statics to the QoSBlockchain distributed database. Figure 32 illustrates the designed monitoring architecture.

Figure 32 – Traffic Monitoring Architecture.



Source: The Author, 2025.

The monitoring method first step starts in the source network domain, when the QoS monitoring rules are created and then expired. Then, the FLOWPRI-SDN of that domain starts collecting the next 20 packets, to analyze calculate the QoS metric values, and marking the packets DSCP field, before sending to the next domain.

In the destination network domain for that flow, the FLOWPRI-SDN collects the marked packets and computes their QoS metric values. Then, it keeps that log and waits for the source network domain's FLOWPRI-SDN to inform its metrics. The communication between them occurs by ICMP. When the FLOWPRI-SDN gets the metrics collected by the source domain, it computes the delay and loss for that flow and send a transaction to the corresponding QoSBlockchain. To finish, the FLOWPRI-SDN sends the data received in the ICMP to the destination host of the flow.

The hosts participate in the QoS monitoring if they run the monitoring daemon application. This application uses *tcpdump* to filter marked packets to calculate the QoS metrics using the *Scapy* tool. Then, after receiving the metrics collected in the source of a flow, it computes the delay and loss for the flow and creates a transaction to the QoSBlockchain for that flow.

All QoS flows registered in the QoSBlockchain represent the effort of the network and clients to improve the network quality. Hosts or framework domains have nothing to gain lying to the blockchain. If the host lies about receiving QoS, then it will

be charged for a service it doesn't receive. But if they lie about not receiving, then the network domains will stop providing QoS. On the other hand, if the network domains lie about providing QoS, the hosts will notice on the measurement phase, and they can be sued later on.

4.5 REQUIREMENTS AND RESTRICTIONS

In this section, we summarize some requirements and restrictions around the FLOWPRI-SDN implementation. This framework has multiple modules to implement bandwidth reservation. However, the routing system is assumed to be manually configured or that an external API has configured it. There are no modules implemented to guarantee the delay restrictions of a flow, we assume that all routes are the best chosen in the scenario.

It is assumed that all the domains of any route for a flow, implements the FLOWPRI-SDN and work in cooperation. Also, it is assumed that the routes does not change during the activity time of the flow that had the FRED accepted. Although, it can change after the flow expire for idle timeout or hard timeout. The FLOWPRI-SDN QoS system is not able to sense the route changes and sense that are incompatible domains in the route, that are not managed by a FLOWPRI-SDN controller.

The FLOWPRI-SDN guarantees that if all domains accepted a flow, it will be forwarding with resource reservation. However, sometimes the FLOWPRI-SDN is unable to provide the QoS and the flow is handled as best-effort.

In the QoS monitoring and traffic classification, occurs the capture of 20 packets of the flows. It can overload the controller, so the network domain must be well configured, with multiple FLOWPRI-SDNs if needed.

Also in the QoS monitoring model, the hosts must run a daemon to filter packets for monitoring. This daemon is not default for any host or operational system. It means, it is an invasive solution that must be installed to work properly.

Another limitation of the FLOWPRI-SDN is the amount of QoSBlockchains that are created. Every pair of end network domains keeps a QoSBlockchain. These blockchains reserve a certain amount of memory and ports to run the hyperledger sawtooth core. It can be a restriction in some scenarios.

4.6 CHAPTER CONSIDERATIONS

In this chapter, the existing components of the FLOWPRI-SDN 2.0 framework were presented, along with the improvements and new components proposed for this

work. With these efforts, the aim is to foster collaboration among autonomous systems that comprise the Internet to implement multi-domain QoS as a service.

The FLOWPRI-SDN framework features a queue configuration that isolates applications traffic flows based on their time requirement characteristics (delay and jitter) into two QoS classes: Real-Time and Non-Real-Time. These classes support different priority levels and share bandwidth between them through the flow admission and allocation mechanism called G-BAM.

The G-BAM performs link-by-link bandwidth allocation in a prioritized manner, where flows with higher priority are given allocation preference. Therefore, each flow undergoes shaping according to its requirements' description (FRED), ensuring that QoS flows are managed to prevent resource contention.

The FRED is obtained from the traffic classification method, that uses Random Forest as the ML model. The ML classifier collects packets from a traffic flow and extracts its statistical features to differentiate its application and map its QoS requirements.

The inter-domain communication is performed using ICMP packets. These packets can carry FREDs (accepting or rejecting) and flow monitoring data. Only end-domains extract the FRED to reserve bandwidth, whereas, backbone domains implement the backbone G-BAM.

Finally, it was proposed to use blockchain as a distributed database to maintain a record of the timeline of the monitored QoS, performed by edge entities (frameworks and hosts). This component aims to enhance the reliability and accountability of the provided QoS. Thus, this component can encourage autonomous systems domains to collaborate on resource reservation without interfering with individual routing policies.

5 EXPERIMENTS

In this Chapter, experiments are performed to analyze the aspects around the traffic classification model. In the first experiment, two data models are evaluated, alongside some ML models and two sample size. In the second experiment, a traffic classification architecture is evaluated, testing various scenarios and sample sizes.

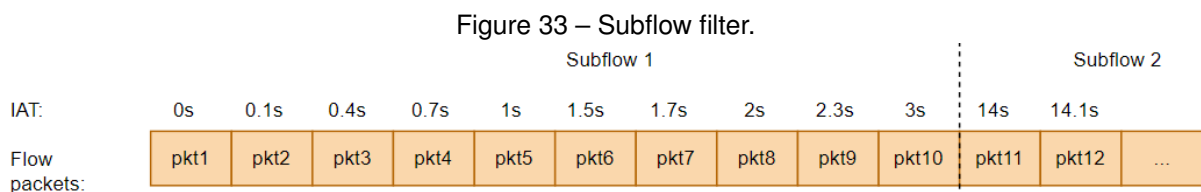
5.1 STREAMING DATA VS BATCH DATA

There are numerous ML algorithms in the literature. Although some are supervised, semi-supervised, or unsupervised, they can also be categorized based on the type of input data as batch-based or streaming-based. Similarly, packet flows can be interpreted as either batch or streaming. Therefore, comparisons have been made between batch-based and streaming-based supervised algorithms in the paper (JR; PARPINELLI; FIORESE, 2023) and the main results are presented in this section.

Three batch-based supervised ML classical algorithms are chosen: Decision Tree, Random Forest and Support Vector Machine (SVM). To represent streaming-based supervised ML, Adaptive Random Forest and Hoeffding Tree were chosen.

The ISCX-VPN-NonVPN-2016 dataset was chosen to train the machine learning models (UNB, 2016). This dataset consists of real network packet captures with a wide variety of labeled application types. Among the captured applications are Skype, Facebook, YouTube, and Netflix. Table 7 shows the packet and flow distribution and the application labels. Each capture is stored in a *.pcap* file format, which facilitates packet processing.

The tool called Scapy was used to process the traffic capture files (BIONDI, 2023). For each application, individual flows were extracted using the 5-tuple information (source IP, destination IP, source port, destination port, transport protocol). From these flows, packets were further extracted into sets where the inter-arrival time between packets did not exceed 10 seconds (subflow). It represents the timeout for the classification. Figure 33 helps to understand what is a subflow.



Source: The Author, 2024.

Table 7 – Division of flows and packets in the experiment's database.

Class of Service	Applications	Flows	Pkts
Audio Real-Time	VoipBuster,Skype, Hangouts	5588	796.964
Static Audio	Spotify	176	26.812
Video Real-Time	Skype,Hangouts	580	978.291
Static Video	YouTube,Netflix	657	370.356
Chat	Facebook,Hangouts, ICQ,AIM,Skype, GmailChat	4363	189.869
Data Download	Bittorrent,ftps,sftp, scp	758	589.583
Data Upload	ftps,sftp	135	200.425
E-mail	Gmail	1400	96.660

Source: (JR; PARPINELLI; FIORESE, 2023).

From the subflows, two scenarios were generated. In the first scenario, samples of 10 packets were processed, while in the second scenario, 30 packets were used. In each scenario, the database was processed for batch-based and streaming-based ML algorithms. The differences between the two datasets are that in batch-based all the sample is processed into one set of features, whereas in the streaming-based model each packet computes one set of features. Table 6 in Section 4.2 shows the labels defined for each sample of packets.

The datasets were balanced. Feature sets from larger classes were randomly removed so that they all matched the same amount:

- Streaming-base with 10pkts: 1770 sets per class.
- Streaming-base with 30pkts: 555 sets per class.
- Streaming-base with 10pkts: 18035 sets per class.
- Streaming-base with 30pkts: 17763 sets per class.

The scenarios were tested using a computer running Arch Linux 64-bit, with an AMD Ryzen 5 5600 processor and 20GB of RAM, utilizing a single thread. The batch machine learning models used were imported from the Python scikit-learn library. For streaming machine learning models, the Python scikit-multiflow library was imported, which implements models based on the Massive Online Analysis (MOA) library.

The selected metrics for model analysis were accuracy, precision, recall and F1-score. Precision indicates how many of the classifier’s identifications were correct. Accuracy measures how often the classifier correctly identifies a data point. Recall represents the rate at which the model correctly predicts how many real positive cases. Lastly, the F1-score is a harmonic average between precision and recall (AGRAWAL, 2023).

After running 5 iterations in each scenario using K-fold cross-validation, with $K = 5$. In the cross validation testing method, the entire dataset is separated into K blocks of the same length, then the training and testing phase is run K times, choosing one of the blocks for testing and the rest for training. Table 8 presents the comparison among the learning models for the scenario with samples of 10 packets. As evident, the batch-based Random Forest model achieved the best metrics and exhibited a low standard deviation across the cross-validation iterations. Among the streaming-based models, Adaptive Random Forest performed better than its counterpart, the Hoeffding Tree.

Table 8 – ML models comparison table (10pkts).

	ML Model	Precision	Accuracy	Recall	F1-Score	Standard Deviation Precision
Batch	Decision Tree	0.97	0.97	0.97	0.97	0.0039
	Random Forest	0.98	0.98	0.98	0.98	0.0045
	SVM	0.9	0.9	0.9	0.9	0.0074
Streaming	Hoeffding Tree	0.75	0.78	0.76	0.72	0.11
	Adaptive Random Forest	0.88	0.85	0.86	0.85	0.024

Source: (JR; PARPINELLI; FIORESE, 2023).

Table 9 comprises the comparison among the ML models for samples of 30 packets. In these scenarios, batch datasets shrink by a factor of 3 because more packets are used to extract feature sets, resulting in fewer data entries. The batch-based models experienced a slight decrease in overall performance, particularly in the cases of Decision Tree and Support Vector Machine (SVM), while the Random Forest model remained unchanged.

On the other hand, in streaming-based models, the dataset size changes little, but the set of features for each sample increases by a factor of 3, leading to the analysis of more feature sets from the same flow. As a result, the models showed improved

overall performance, especially the Hoeffding Tree model, which increased precision by 16%.

Table 9 – ML models comparison table (30pkts).

	ML Model	Precision	Accuracy	Recall	F1-Score	Standard Deviation Precision
Batch	Decision Tree	0.95	0.95	0.95	0.95	0.013
	Random Forest	0.98	0.98	0.97	0.98	0.0048
	SVM	0.73	0.71	0.71	0.67	0.026
Streaming	Hoeffding Tree	0.91	0.92	0.91	0.90	0.015
	Adaptive Random Forest	0.91	0.92	0.92	0.91	0.025

Source: (JR; PARPINELLI; FIORESE, 2023).

For batch data classification models, training and testing datasets occur nearly instantaneously (less than 1 second). However, for the 10-packet subflow dataset, the Hoeffding Tree model averaged 9 seconds for training in each cross-validation iteration and 4.17 seconds for testing. In contrast, the Adaptive Random Forest model averaged 188.3 seconds for training and 22.9 seconds for testing on the same dataset.

For the 30-packet dataset, the Hoeffding Tree model averaged 7.9 seconds for training in each iteration and 4.15 seconds for testing. Conversely, the Adaptive Random Forest model required 176.3 seconds for training and 23.56 seconds for testing.

These runtime illustrate that while batch-based models are extremely fast, streaming-based models, particularly the Adaptive Random Forest, require significantly more processing time due to their continuous adaptation to incoming data streams. Therefore, in these tests, the Random Forest classifier got the best general results. However, in these tests, the models were not fine-tuned, that can have influenced the results in some aspects.

5.2 REFINING THE TRAFFIC CLASSIFICATION ARCHITECTURE

The first traffic classification experiment showed how ML models behaved in scenarios where 10 and 30 packets for each flow were collected for QoS classification. However, the classes of the classification do not reflect the QoS classes that the FLOWPRI-SDN needs to generate the flow's FRED.

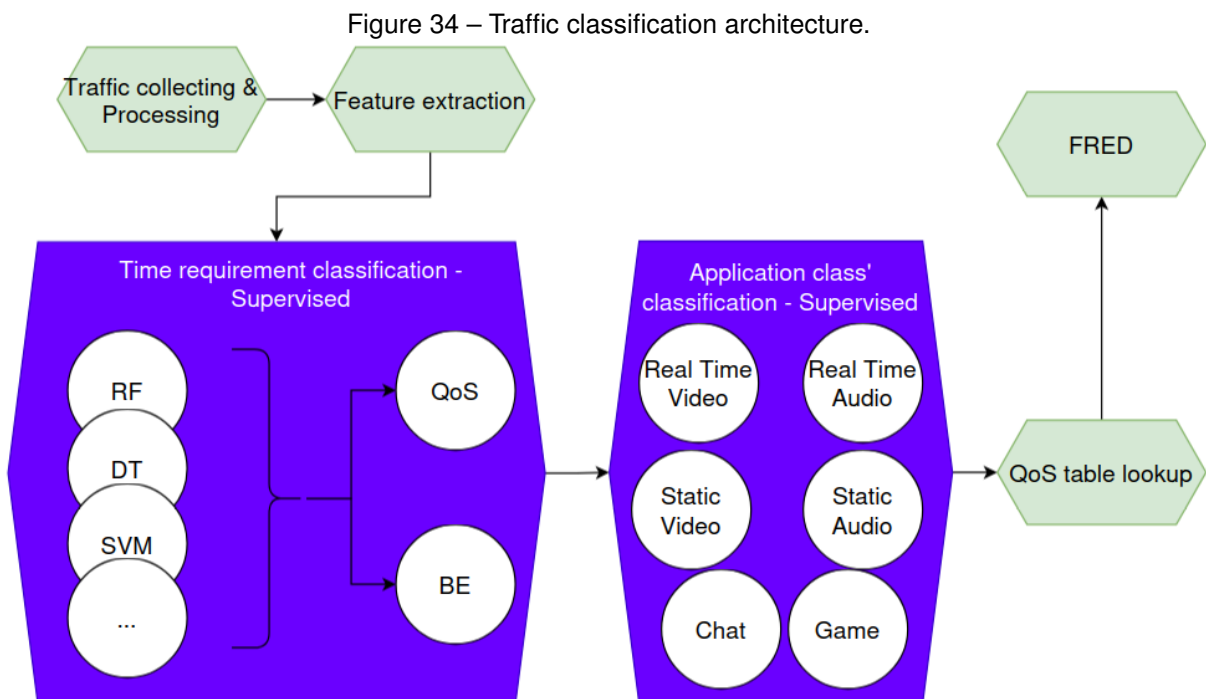
In FLOWPRI-SDN 2.0, the contract model was replaced for the traffic classification. It means that, the hosts cannot send QoS contracts, all the QoS discovery is

done by the controller. That is one reason to change the idle-timeout to 2 seconds, instead of 10 seconds as experienced before.

There are more aspects to consider in this experiment. No hyperparameter tuning was performed in the previous one, that can bring differences on the performance (not always though). Also, the inter-arrival time was 10 seconds, that reflected the idle-timeout in FLOWPRI-SDN 1.0 for the contracts. However, idle-timeouts of 10 seconds hold the bandwidth for too long in the switches, when a flow ends. It can also make the controller less sensitive to QoS changes.

This experiment is intended to test the idle-timeout of 2 seconds and more scenarios for traffic classification. In the practical context, it means that more flows will be tested, and the datasets are bigger. For this experiment, the ML models implemented to comparison are: Random Forest, Decision Tree, SVM, XGBoost, Adaptive Random Forest, and Hoeffding Tree. After the classification process, the labels obtained are used to fill the QoS fields of the FRED, that represent the estimated the QoS requirements, such as: bandwidth, loss, delay, and jitter.

Thus, we proposed a traffic classification architecture of three phases. The Figure 34 presents the steps of each phase. In the first phase, the flows are categorized in best-effort and QoS. In the second phase, the flows are categorized in application classes. The application classes can be: real-time video, real-time audio, static audio, static video, chat (textual, voice and video), and gaming. These classes represent the big picture of multimedia traffic accessed on the Internet.



Source: The Author, 2025.

Table 10 – Flows distribution

Class of Service	Application Class	Application Type
Real-Time	Audio, Video, Online Chat, Gaming	Voipbuster, Skype, Hanghouts, CS2, Online Chess, Skype, Facebook, ICQ
Non-Real-Time	Audio, Video	Spotify, YouTube, Netflix, Twitch
Best-Effort	www, donwload, upload, e-mail	Bittorrent, Skype, Gmail, generic

Source: The Author, 2025.

Inside the application classes, there are application types. They represent some application names, that are accessed by hosts, such as: YouTube, Vimeo, Skype, Facebook and others. In the third phase, the flows are mapped into QoS, using the QoS table presented in Table 6 to get the best quality for the flow.

5.2.1 Traffic collection and Processing phases

The ISCX-VPN-NonVPN-2016 database of traffic captures is used in this experiment too (UNB, 2016). As explained before, the capture files (*.pcap*), are separated by application type. All the files were processed extracting the flows, storing them into new *.pcap* files. Excluding files with less than 1 Kb and flow with protocols like: DNS, ARP and LLDP.

The Table 10, shows the service classes and the application classes, that are the goals of the classification. In the previous experiment, for simplicity, many files were excluded from the data processing, keeping only the biggest ones. Another difference for this experiment is that, online chatting is considered a real-time application, whereas, download, upload, e-mail, and WWW, is considered best-effort traffic.

All flows' files are processed with *Scapy*. We generate datasets for 10, 30 and 50 packets from subflows that have inter-arrival time lesser than 2 seconds (idle timeout for the QoS rules). The amount of packets for the classification is an important parameter for the final solution overhead. The smaller the sample, the faster the classification will finish and the QoS rules will be created. The scenarios are: full flow one-way, full flow two-ways, one-way, one-way TCP only and two-ways TCP only.

In the full flow one-way scenario, the *.pcap* files are separated into forwarding and receiving traffic, then all packets of each flow are processed into one set of features. In the full flow two-ways, all the *.pcap* file of a flow is processed into a set of features. The other scenarios' datasets are processed by separating the flows into subflows with inter-arrival time of less than 2 seconds for the cases of 10, 30 and 50

Table 11 – Table of features part 1.

Number	Short	Description
1	Server port	Port number at server
2	Client port	Port number at client
3-9	Total pkts IAT stats	Min, max, var, q1, q3, mean, med
10-16	Total Ethernet pkts stats	Min, max, var, q1, q3, mean, med
17-23	Total IP pkts stats	Min, max, var, q1, q3, mean, med
24-30	Total TCP/UDP pkts stats	Min, max, var, q1, q3, mean, med
31-32	Pkts A->B, B->A	Number of packets seen in one direction
33-152	Total TCP specific	Flags, retransmission, acks, window
153-159	A->B Ethernet pkts stats	Min, max, var, q1, q3, mean, med
160-166	A->B IP pkts stats	Min, max, var, q1, q3, mean, med
167-173	A->B TCP/UDP pkts stats	Min, max, var, q1, q3, mean, med
174-180	B->A Ethernet pkts stats	Min, max, var, q1, q3, mean, med
181-187	B->A IP pkts stats	Min, max, var, q1, q3, mean, med
188-194	B->A TCP/UDP pkts stats	Min, max, var, q1, q3, mean, med
195-201	A->B IAT pkts stats	Min, max, var, q1, q3, mean, med
202-208	B->A IAT pkts stats	Min, max, var, q1, q3, mean, med
209	Time since last connection	Time since the last connection between these hosts

Source: (MOORE; ZUEV; CROGAN, 2013).

packets. For the two-ways and TCP scenarios, the tool *tcptrace* is used to extract the features from the packets. This tool was developed in 2003 by Shawn Ostermann and can analyze multiple features of TCP flows (OSTERMANN, 2003).

Moore et al. 2013, presented a great list of discriminators for flow-based classification, that totalizes 249 features. They are summarized in Tables 11 and 12. Basically, the discriminators can be classified into inter-arrival time, packet length and TCP specific.

Table 12 – Table of features part 2.

Number	Short	Description
210	Bulk transitions	Bulk = time when there are more than three successive packets in the same direction without any packets carrying data in the other direction
211	Time spent in bulk	Amount of time spent in bulk transfer mode
212	Duration	Connection duration
213	% bulk	Percent of time spent in bulk transfer
214	Time in idle	The time spent idle
215	% idle	Percent of time spent idle
216	Effective bandwidth	Effective Bandwidth based upon entropy
217	A->B effective bandwidth	Effective Bandwidth based upon entropy A->B
218	B->A effective bandwidth	Effective Bandwidth based upon entropy B->A
219-249	FFTs	Arctan of the top-ten frequencies IAT ranked by the magnitude of their contribution

Source: (MOORE; ZUEV; CROGAN, 2013).

The *tcptrace* tool can calculate most of the features for TCP statistics (33-152 and 209), and the remaining are calculated using *Scapy*. However, some features are impractical or scenario specific, and are removed, such as: 1, 2, 209, 210, 211, 213, and 219-249. Also, the features that depend on the IP and Ethernet headers (10-23, 153-166, and 174-187) were removed, so that the classification results can extend to IPv4 and IPv6 flows.

After generating the datasets of features for each scenario, they are normalized using the same values as parameter for *value/maximum*, and it is performed data balancing by under-sampling. In the phase 1 of the classification, it is intended to separate QoS traffic from best-effort traffic. First, the best-effort total amount of data is divided half for WWW + E-mail and half for download and upload traffic. Second, it is calculate the smaller QoS application class. Then, it is decided which is the smaller, the application class amount multiplied by 6 (the total application classes) or the half calculated for best-effort times 2 (the two parts), and the value is stored in the variable *N*. Deciding the smaller number, each application class data and the two parts for best-effort are shuffled, and the first $N/numberOfClasses$ entries for each class is stored in a new balanced dataset.

The datasets for the phase 2 is intended to classify the data into one application class. There are six application classes: Real-time video, real-time audio, static video, static audio, online chat, and gaming. The balanced datasets are created by finding the application class with the smaller amount of data (let's call the value as *M*). Then, the

data is shuffled, and the balanced dataset is the concatenation of the first M entries of each class.

5.2.2 Training Phase

The ML models have multiple parameters to configure. There is a default value pre-configured for them, but some models feel the performance improvement when they are adjusted for the dataset. This procedure of finding the best hyperparameter setting for a machine learning model is often known as hyperparameter tuning.

One of the ways of performing hyperparameter tuning is manually. In this case, a set of parameters is chosen and a set of values are tested in turns. The model is trained with a set of values in each turn, comparing to the previous one, to determine which values result in the best performance. However, hyperparameter Tuning doesn't work all the time, and sometimes the default hyperparameters are also considered to be the best estimators.

Some parameters that can affect the ML models were chosen for tuning. Here is the list of parameters tuned for each model using grid search:

- SVM: C (1000), kernel (rbf), and gamma (1).
- Random Forest: Number of estimators (25), max features per tree (log2), max depth (6), and max leaf nodes (6).
- Decision Tree: Max depth (10), min samples split (2), and min samples per leaf (4).

5.2.3 Results

In order to analyze the performance of the ML models for identifying best-effort flows and QoS flows, some tests are implemented. All the scenarios were processed using the K-Fold scheme. In this method, the dataset is separated in 5 parts, and it runs 5 turns. In each turn, one different part of the dataset is used for testing, and the remaining, as training.

Table 13, shows the results for the first Phase of the traffic classification architecture, the value on the left is the accuracy, and on the right is the precision. In this case, it is noted that, the dataset used for hyperparameter tuning in the Random Forest model was not efficient, but it boosted the performance of the Decision Tree classifier. In this scenario with only two classes, the amount of packets showed not so determinant for the performance as the protocol isolation.

Datasets that isolated TCP flows from the UDP ones, got overall better performance. These datasets, also have more features, that helped to reach higher precision. It is important to notice, that using all the flow to compute one set of features does not results in the best performance, as can be seen in the one way and two-ways datasets of total flows. For the ML models and the context of traffic classification, the amount of data is determinant. That is the reason for datasets with short samples to perform well.

Table 13 – Results for Phase 1

Scenarios	Random Forest	Decision Tree	SVM	XGBoost
AB 10pkts	0.8;0.81	0.92;0.92	0.71;0.71	0.95;0.95
AB 30pkts	0.83;0.84	0.94;0.94	0.66;0.66	0.96;0.96
AB 50pkts	0.85;0.85	0.95;0.95	0.53;0.53	0.96;0.96
AB TCP 10pkts	0.88;0.88	0.98;0.98	0.67;0.68	0.99;0.99
AB TCP 30pkts	0.89;0.89	0.97;0.97	0.53;0.53	0.99;0.99
AB TCP 50pkts	0.9;0.9	0.98;0.98	0.51;0.25	0.99;0.99
2Ways 10pkts	0.87;0.87	0.97;0.97	0.69;0.69	0.99;0.99
2Ways 30pkts	0.89;0.9	0.97;0.97	0.68;0.68	0.99;0.99
2Ways 50pkts	0.88;0.88	0.96;0.96	0.52;0.26	0.98;0.98
AB total	0.73;0.74	0.87;0.87	0.53;0.58	0.92;0.92
2Ways total	0.82;0.82	0.93;0.93	0.48;0.42	0.98;0.98

Source: The Author, 2025.

In the second Phase of the classification architecture, the database of flows was processed into six classes: real-time video, real-time audio, static video, static audio, gaming, and chat. Each of these classes is a group of specific application traffics, such as YouTube and Vimeo. This way, it may be harder to classify similar flows into different classes. However, the second Phase got similar results to the first table.

Table 14, shows that scenarios with mix of TCP and UDP flows depend on the quality of the value of the features. Overall, there is a big difference in accuracy between scenarios of samples of size 10 and 50 packets. Even though, the 10 packets' scenarios have more data, the quality is poor.

Nonetheless, the datasets of isolated TCP flows still performed better. In these scenarios, the sample size did not affect the performance as in the scenario with mixed TCP and UDP flows.

In terms of ML models, only Random Forest and Decision Tree had hyperparameter tuning. However, sometimes the default parameters are better suited, and the parameters are very sensitive to the datasets. As the hyperparameter tuning was performed using only one dataset, it can be the reason for the poor performance in the Random Forest, but also the improvement in performance in the Decision Tree.

That said, XGBoost classifier has shown the best performance, beating the other models in almost all scenarios, and should be considered as the best option for

Table 14 – Results for Phase 2

Scenarios	Random Forest	Decision Tree	SVM	XGBoost
AB 10pkts	0.76;0.77	0.88;0.9	0.76;0.77	0.9;0.91
AB 30pkts	0.8;0.81	0.92;0.93	0.71;0.75	0.93;0.94
AB 50pkts	0.82;0.84	0.93;0.93	0.63;0.57	0.95;0.95
AB TCP 10pkts	0.83;0.69	0.99;0.99	0.7;0.76	0.99;0.99
AB TCP 30pkts	0.82;0.62	0.99;0.99	0.41;0.26	0.99;0.99
AB TCP 50pkts	0.82;0.69	0.99;0.99	0.31;0.21	0.99;0.99
2Ways 10pkts	0.8;0.7	0.98;0.99	0.76;0.83	0.99;0.99
2Ways 30pkts	0.83;0.71	0.99;0.99	0.69;0.62	0.99;0.99
2Ways 50pkts	0.84;0.72	0.98;0.98	0.46;0.40	0.99;0.99
AB total	0.5;0.44	0.65;0.65	0.18;0.14	0.73;0.75
2Ways total	0.69;0.59	0.91;0.92	0.16;0.19	0.94;0.94

Source: The Author, 2025.

the traffic classification architecture.

5.3 FLOWPRI-SDN 2.0 OVERHEAD

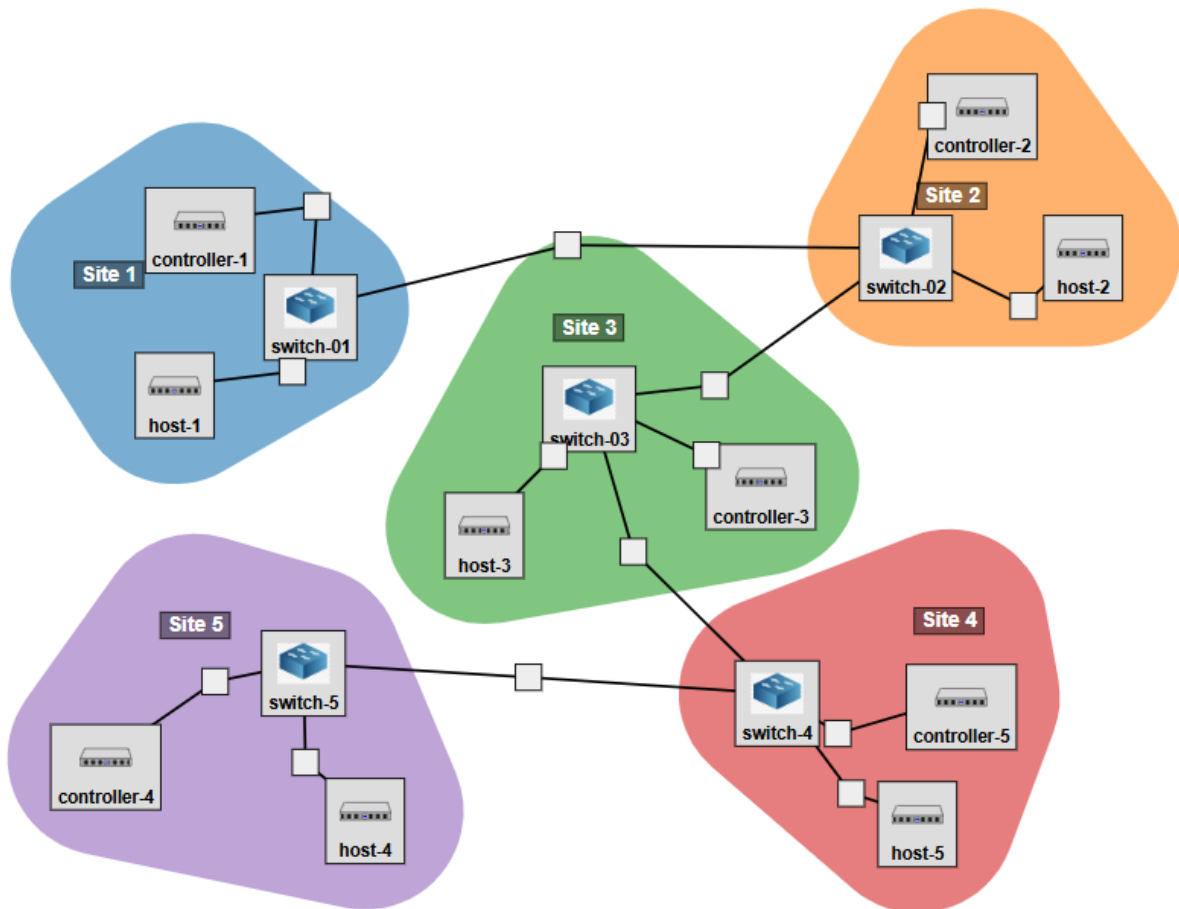
In this section, we introduce the experiments performed with the FLOWPRI-SDN 2.0 and its components. It is intended to analyze the overhead in time motivated by creating the QoS rules, performing traffic classification, traffic monitoring and managing the QoSBlockchains.

To explore that, it is configured one topology with multiple domains, each one with one SDN controller (FLOWPRI-SDN 2.0) and one host, that is both, conventional host and management host, as it is shown in Figure 35. One OVS switch is configured for each domain, interconnecting the domains. The same scenario is used in all experiments.

The links speed are set to $50Mbps$, that are shared with HTB queues as soon as the the switches connect to the FLOWPRI-SDN 2.0 controller of their domains (switch configuration step). Each interface delays the packets for 5ms to simulate the delay of the links. For instance, a ping message between H1 and H2 takes almos 20ms to arrive from one side to the other, even without FLOWPRI-SDN 2.

The experiments are designed to focus on the processing time and the number of packets exchanged by the proposed solutions. Each experiment was repeated 20 times to calculate the average values. Therefore, a network simulator was built using the Linux IP NETNS tool to create the topology on a single machine (ROSEN, 2016).

Figure 35 – Network Topology for the experiments.



Source: The Author, 2025.

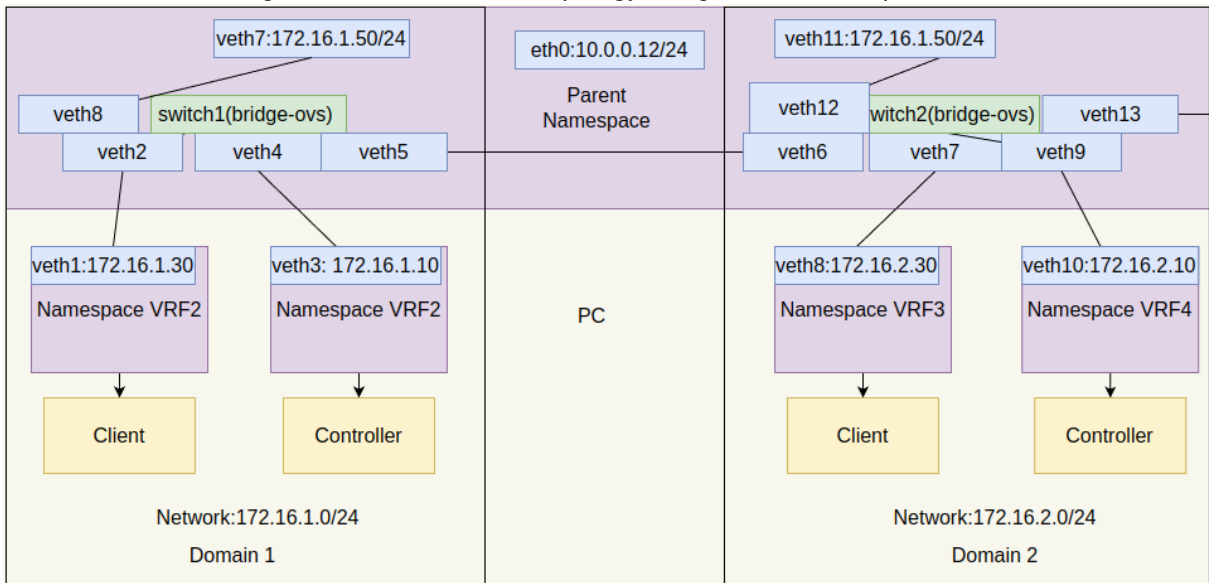
5.3.1 Building the Topology

The IP-NETNS tool is a Linux application for managing virtual network namespaces. A network namespace is logically another copy of the network stack, with its own routes, firewall rules, and network devices. By default a process inherits its network namespace from its parent. Initially all the processes share the same default network namespace from the init process (ROSEN, 2016; KERRISK, 2013).

The OVS instances runs in the parent namespace. But, each host and controller is separated in one individual network namespace. Then, a pair of virtual network interfaces is created for them. One interface is moved to the network namespace and the other stays at the parent namespace to connect with the OVS instance of the domain. A pair of virtual network interfaces is created to connect OVS instances, connecting their domains. Therefore, the topology can be seen better in the Figure 36, that shows the namespace distribution.

With this configuration the hosts are network separated as the topology demands. Although they are not in different machines, it also means they run the same hardware, and the processing time of for the FLOWPRI-SDN 2.0 is not influenced by

Figure 36 – A slice of the topology design, create with ip-netns.



Source: The Author, 2025.

different machine configuration. For the record, the machine used in the experiments has a Ryzen 5 5600 processor, 19Gb of Ram, and runs over Ubuntu 24.04.2LTS. Therefore, all of the experiments presented in this chapter can be replicated, following the steps and configurations of the link provided in Appendix A.

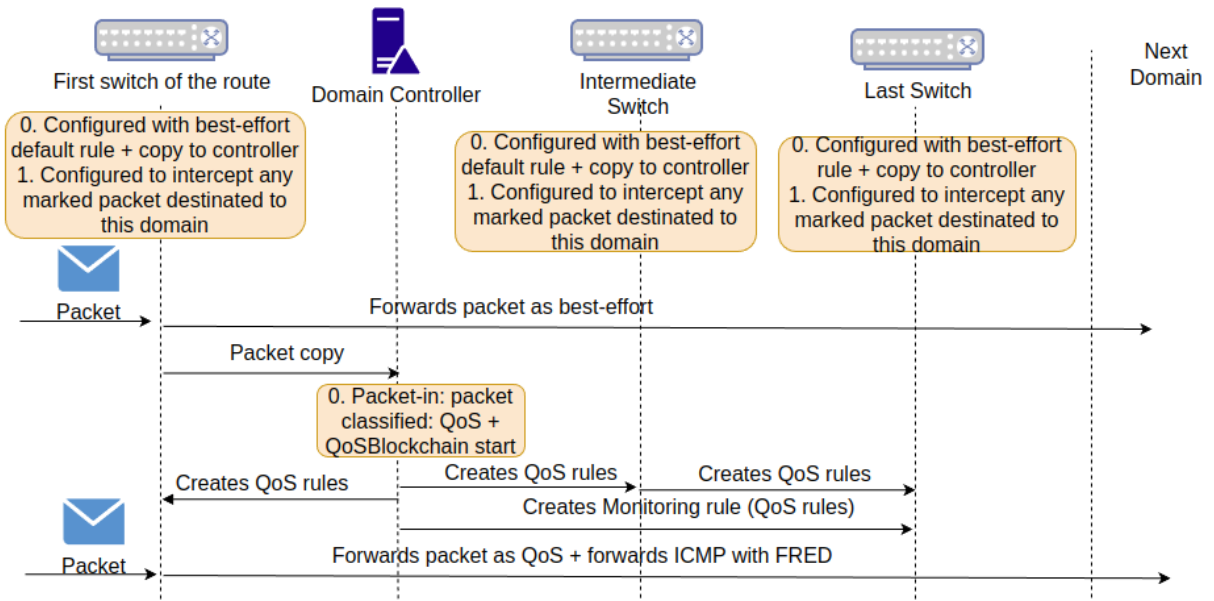
5.3.2 Flow Classification Overhead

Using the *iperf* tool, two hosts from different domains begin to communicate. During this period, the first packet reaches the source domain gateway and is processed by FLOWPRI-SDN 2.0. As a result, a best-effort rule is created to forward the flow's packets to the next domain, while a copy of those packets is sent to the source domain controller to initiate the classification phase.

The behavior continues until the tenth packet copy reaches the source domain controller. At that point, the flow is classified using a Random Forest model to process the FRED according to the flow's QoS requirements. If the flow is identified as best-effort, the rule is updated to stop forwarding packet copies to FLOWPRI-SDN 2.0. However, if the flow is classified as QoS, the rules are updated accordingly, creating a meter rule and assigning the appropriate queue. To complete the setup, an ICMP packet (type 15 or 139) is sent toward the flow's destination, allowing the next-hop domains to configure their rules as specified in the FRED. These processes are presented in Figure 37

The Figure 38 shows the graph for the time at which the first packet from a host reaches FLOWPRI-SDN 2.0 and the flow is classified as QoS. Following this, the corresponding rules are created and the ICMP packet announcing the flow's FRED is

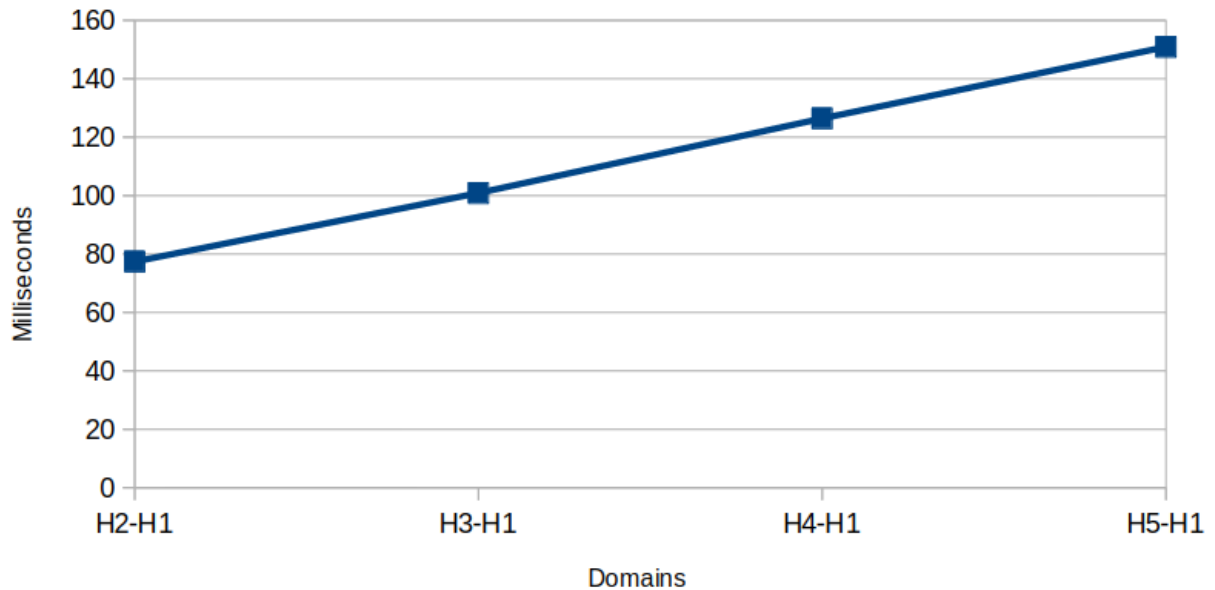
Figure 37 – Flow classification diagram.



Source: The Author, 2025.

sent. All of these steps are accounted for in the chart.

Figure 38 – Time from the first packet until QoS.



Source: The Author, 2025.

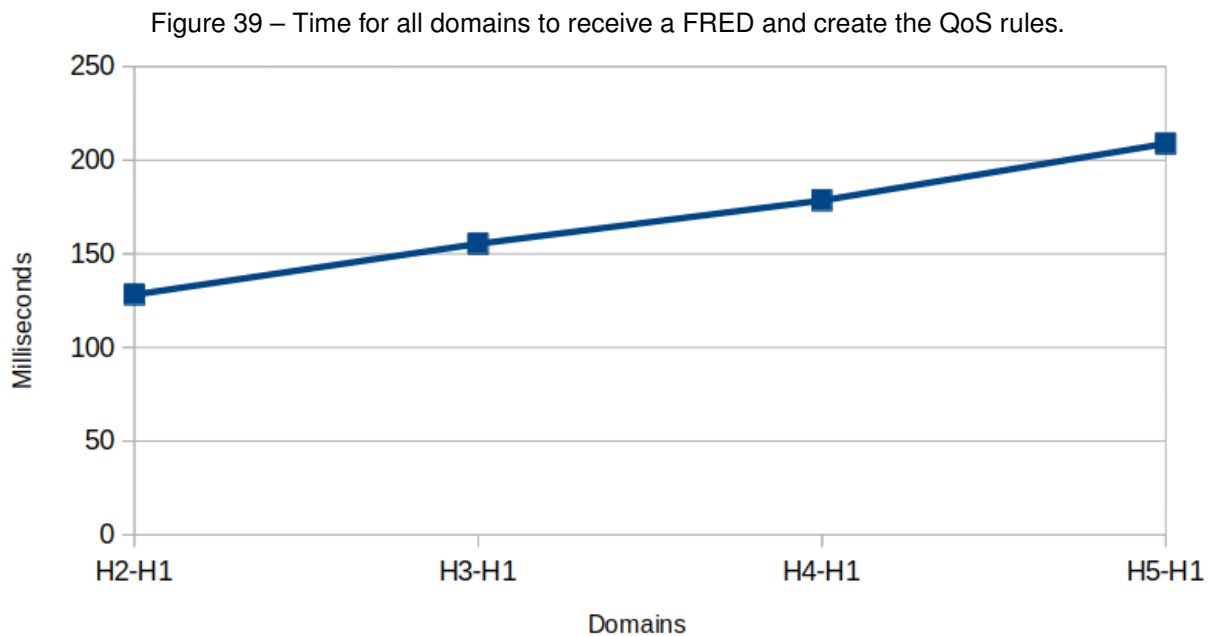
However, the time for the QoSBlockchain setup was not counted in the charts. It takes around 500ms and 600ms to setup, but as it runs as a different thread, and the monitoring rules will only be activated many seconds after that, it was not included in the captured time.

In this experiment, every hop counts 10ms because of the topology design. However, the packets are copies, it means the communication still goes as best-effort

while the classification is happening. It takes a little time to get 10 packets to make the classification.

Taking the first scenario as example, H2 forwarding packets to H1, 80ms was all it took to change a best-effort flow into a QoS one. From this value, 20ms is implied by the links, and only 60ms for the processing steps, 20ms in the first domain and 40ms in the second. The other scenarios keep the time consistent, increasing basically the link delay.

In the Figure 39, it was monitored the time since the first packet was sent, until all domains receive the ICMP announcing the flow's FRED and the rules were created. The time was captured in the first controller and the last of the domain.



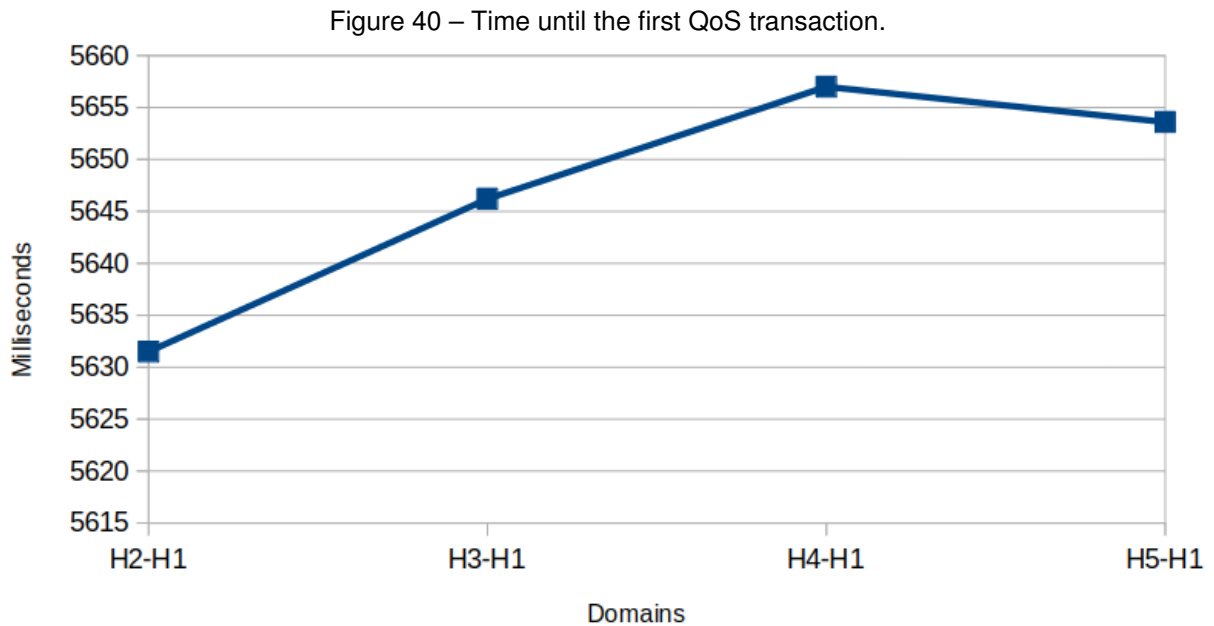
Source: The Author, 2025.

The time adds only around 40ms compared to the first scenario. This indicates that inter-domain communication was not time-intensive, and that link delay had a much greater impact than the QoS mechanisms of FLOWPRI-SDN 2.0.

Finally, the time from the first packet sent until the destination host creates the first QoS transaction was monitored. This occurs after the monitoring rules are established. For this experiment, a monitoring timeout of 5 seconds and a hard timeout of 10 seconds were configured, meaning the QoS flow is monitored every 5 seconds.

Figure 40 presents the timing for this step. During the monitoring phase, the FLOWPRI-SDN 2.0 in the source domain monitors and marks the packets. It then sends the monitoring metrics to the FLOWPRI-SDN 2.0 of the destination domain, which also monitors the packets and uses the received metrics to compute a transaction (this step was not observed in the experiment). Afterward, it forwards the metrics to

the destination host, which performs the same computation and creates a transaction as well, completing the QoS cycle of the FLOWPRI-SDN 2.0 components.



Source: The Author, 2025.

After the 5-second timeout, the destination hosts calculate the delay, jitter, packet loss, and bandwidth based on the monitoring packets. Once again, it is evident that the farther apart the hosts are, the more significant the link delay becomes. It takes approximately 600ms from the moment the source host sends the first packet until it is monitored and a QoS transaction is sent.

5.4 CHAPTER CONSIDERATIONS

In this chapter, multiple ML algorithms for data classification were compared. For the first experiments, there are considered two classes of data to be consumed by ML models: streaming-based and batch-based. In the streaming-based model, continuous data streams are fed into the classifier, which produces an output for each input. In the batch-based model, a sample needs to be processed in one go as an input, which is then consumed to generate a classification.

Furthermore, there are several supervised traffic classification algorithms that fit into the two categories of data classes described, batch and streaming data. To the best of our knowledge, there were no previous works comparing the results for these two data models. Therefore, initial comparative tests were conducted to observe the strengths and weaknesses of each algorithm.

To achieve this, a dataset with labeled traffic captures from various types of network applications was used. This dataset was processed into subflow samples of

10 and 30 packets, generating sets of statistical features. The dataset was then labeled according to the application requirements and balanced to help to prevent algorithms from assigning all cases to the most common outcome, and can improve accuracy rates.

Thus, the cross-validation ML evaluation technique was employed. In this model, the dataset is divided into equal fractions (5 folds), where iterations (also 5), are performed. One fraction is used for testing, while the remaining are used for training. As a result, the Random Forest model proved to be the fastest to execute and achieved the highest overall accuracy.

In the second experiment, it was proposed a more refined architecture for traffic classification, separating it into two phases. The first phase classifies best-effort flows and QoS flows. It was missing in the first experiment. In the second phase, the QoS application classes were classified. Many scenarios were explored to identify which model and scenario should be considered the most suitable to the architecture. In the end, the amount of data showed to be more determinant than the quality, and that, scenarios with only TCP flows showed to have the best performance, with the Decision Tree and XGB models.

Experiments were also conducted for the FLOWPRI-SDN 2.0 components. A network simulator was implemented using the Linux NETNS tool to represent interconnected network domains. The links were configured with a static delay of 5ms per interface, and the bandwidth was set to 50Mbps.

First, the scenario was presented in which a host sends a packet from a flow, and it is initially forwarded as best-effort until it is classified as a QoS flow by FLOWPRI-SDN 2.0. During this period, the time was monitored to evaluate the overhead introduced by the FLOWPRI-SDN 2.0 mechanisms.

Analyzing the data, each domain takes approximately 20ms to process the 10 packets and apply the classification and QoS rules. Similarly, the inter-domain communication and QoS negotiation phases are not time-consuming. Therefore, it was concluded that FLOWPRI-SDN 2.0 can be a viable option for implementing multi-domain QoS.

6 FINAL CONCLUSIONS

The conventional networks implement only the best-effort model for traffic forwarding. It means that traffic compete for bandwidth, not taking into account their requirements for work, and may cause packet loss, delay, and low bandwidth. In the same context, network domains are interested in managing only the right amount of resources, while providing best services for their clients. Although there are tools available, they are default for network domains to implement resource reservation and synchronizing across the network domains along the route. Similarly, there is a lack of trust between domains to implement end-to-end quality of service, since all must collaborate, and methods for clients to track the services they receive are lacking.

In this study, those gaps were explored. New modules and improvements were developed for the existing FLOWPRI-SDN framework, building the version 2.0. For each challenge, was proposed a module. The resource allocation inside network domains is managed by G-BAM, HTB queues and meter rules. Then, the inter-domain communication is done with ICMP messages in a specific protocol combining a flow requirements descriptor (FRED).

It was proposed an architecture to collect packet samples from flows and classifying traffic using a machine learning (ML) model in the FLOWPRI-SDN 2.0. This model categorizes traffic into an application class (e.g., video-real-time-5Mbps), which enables the generation of the FRED. It was also proposed a passive traffic monitoring that synchronizes the flow's destination host and the FLOWPRI-SDN 2.0 to measure the QoS for the flows and create the transactions in the QoSBlockchain.

In the aspect of thrust among network domains and QoS traceability, it was proposed the QoSBlockchain. It makes every pair of end network domains build a distributed database. One host of the source and destination domains, called management host, also participate as pairs. Therefore, they maintain a register of statistics of QoS for traffic flows.

In FLOWPRI-SDN, clients are part of the architecture. The domains can define management hosts, to be peers of the QoSBlockchains. Also, all the hosts must run a service to monitor traffic, to be able to create transactions in the correspondent QoSBlockchain. Although it can be considered network instrumentation, it is also benefiting for them all.

The components of the FLOWPRI-SDN 2.0 were evaluated in Chapter 5. First, we developed a couple of experiments to evaluate various supervised machine learning models. In this experiment, a dataset of captured flows from various network appli-

cations such as video, audio, and other best-effort ones was used. These flows were processed and labelled to generate sets of features, which were then consumed by the evaluated machine learning models.

Chapter 5 developed a preliminary experiment to evaluate various supervised machine learning models. In this experiment, the dataset ISCX-VPN-NonVPN-2016 was improved with more traffic collected by the authors. The capture files are composed of flows from various network applications such as video, audio, and other best-effort ones were used. These flows were processed and labelled to generate sets of features, which were then consumed by the evaluated machine learning models. With these tests, we could conclude that Random Forest and XGBoost are very efficient models for traffic classification. Also, it is noticed that, the quantity of data is very important for traffic classification, and even using only 10 packets as sample size, these datasets compensate by the amount and compute better precision.

Still in the traffic classification matter, it was explored different scenarios for datasets. The results showed that the amount of data was more important than the amount of packets used in the classification (for the model proposed). In the same way, datasets with only TCP flows got better results than those with mixed (TCP and UDP). In part because they had more features, but also because the dataset became more clean.

In the same chapter, a series of experiments were conducted to evaluate FLOWPRI-SDN 2.0. It was concluded that FLOWPRI-SDN 2.0 takes approximately 20ms to identify a traffic flow as a QoS flow and to announce its FRED to the other domains. Therefore, it introduces minimal overhead relative to the benefits it provides.

BIBLIOGRAPHY

AGRAWAL, S. K. **Metrics to Evaluate your Classification Model to take the right decisions**. 2023. <https://www.analyticsvidhya.com/blog/2021/07/metrics-to-evaluate-your-classification-model-to-take-the-right-decisions/>.

AL-HARBI, A. et al. Towards an efficient resource allocation based on software-defined networking approach. **Computers & Electrical Engineering**, v. 92, p. 107066, 2021. ISSN 0045-7906. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S004579062100080X>>.

AL-JAWAD, A. et al. An innovative reinforcement learning-based framework for quality of service provisioning over multimedia-based sdn environments. **IEEE Transactions on Broadcasting**, IEEE, v. 67, n. 4, p. 851–867, 2021.

ALKENANI, J.; NASSAR, K. A. Network monitoring measurements for quality of service: A review. **Iraqi Journal for Electrical & Electronic Engineering**, v. 18, n. 2, 2022.

ALLEN, S. **ISP Networks: Understanding the Fundamentals**. 2023. Shaun Allen. Disponível em: <<https://www.shaunallen.co.uk/blog/isp-networks>>. Acesso em: 10.6.2024.

AMARAL, P. et al. Machine learning in software defined networks: Data collection and traffic classification. In: **2016 IEEE 24th International Conference on Network Protocols (ICNP)**. [S.l.: s.n.], 2016. p. 1–5.

AZIZ, W. A. et al. Content-aware network traffic prediction framework for quality of service-aware dynamic network resource management. **IEEE Access**, IEEE, 2023.

BABIARZ K. CHAN, F. B. J. **Configuration Guidelines for DiffServ Service Classes**. [S.l.], 2006. Disponível em: <<https://datatracker.ietf.org/doc/html/rfc4594>>.

BADASYAN, N.; CHAKRABARTI, S. et al. Private peering among internet backbone providers. **Economics Working Paper Archive, Series on Industrial Organization, Washington University in St. Louis (WUSTL)**, 2003.

BARKIRE, D. I. et al. Qosflow: Prioritizing flows in software-defined networking for enhanced quality of service. In: **2025 27th International Conference on Advanced Communications Technology (ICTACT)**. [S.l.: s.n.], 2025. p. 1–7.

BASUMALLICK, C. **TCP vs. UDP: Understanding 10 Key Differences**. 2022. Spiceworks. Disponível em: <<https://www.spiceworks.com/tech/networking/articles/tcp-vs-udp>>. Acesso em: 10.6.2024.

Belgaum, M. R. et al. Integration challenges of artificial intelligence in cloud computing, internet of things and software-defined networking. In: **2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)**. [S.l.: s.n.], 2019. p. 1–5.

BHOLEBAWA, I.; DALAL, U. Design and performance analysis of openflow-enabled network topologies using mininet. **International Journal of Computer and Communication Engineering**, v. 5, p. 419–429, 01 2016.

BIONDI, P. **Welcome to Scapy's documentation!** 2023. <https://support.google.com/youtube/answer/78358?hl=en>.

BRADEN, R. T. et al. **A Framework for Integrated Services Operation over Diffserv Networks**. RFC Editor, 2000. RFC 2998. (Request for Comments, 2998). Disponível em: <<https://www.rfc-editor.org/info/rfc2998>>.

CACHEDA, R. A. et al. Qos requirements for multimedia services. In: **Resource Management in Satellite Networks: Optimization and Cross-Layer Design**. [S.l.]: Springer, 2007. p. 67–94.

CACHEFLY. **Discover How Autonomous Systems, ISPs, and ASNs Drive Digital Highways**. 2023. Cachefly. Disponível em: <<https://www.cachefly.com/news/discover-how-autonomous-systems-isps-and-asns-drive-digital-highways>>. Acesso em: 10.6.2024.

CHEN, Y.; FARLEY, T.; YE, N. Qos requirements of network applications on the internet. **Systems Management**, v. 4, 01 2004.

COMMISSION, F. C. **Broadband Speed Guide**. 2013. Disponível em: <<https://www.fcc.gov/consumers/guides/broadband-speed-guide>>.

COMPUTERPHILE. **BGP: Border Gateway Protocol - Computerphile**. 2016. YouTube. Disponível em: <https://www.youtube.com/watch?v=O6tCoD5c_U0&ab_channel=Computerphile>. Acesso em: 10.6.2024.

CRAWFORD, M.; HABERMAN, B. **IPv6 node information queries**. [S.l.], 2006.

DAS, L. et al. Recent aspects of ipv6 on security challenges in iot. In: IEEE. **2023 10th International Conference on Computing for Sustainable Global Development (INDIACom)**. [S.l.], 2023. p. 890–896.

EOM, W.-J. et al. Network traffic classification using ensemble learning in software-defined networks. In: **2021 International Conference on Artificial Intelligence in Information and Communication (ICAIIIC)**. [S.l.: s.n.], 2021. p. 089–092.

ERVASTI, K. A survey on network measurement: Concepts, techniques, and tools. **University of Helsinki**, 2016.

FAUZI, N. F. N. M.; CHIN, T. S.; ZAKI, F. A. M. Implementation of machine learning-based qos traffic classification in software-defined networking. In: **2024 IEEE Asia-Pacific Conference on Applied Electromagnetics (APACE)**. [S.l.: s.n.], 2024. p. 71–74.

FOUNDATION, T. L. **Sawtooth**. 2024. <https://sawtooth.splinter.dev/>.

GAO, L. On inferring autonomous system relationships in the internet. **IEEE/ACM Transactions on Networking**, v. 9, n. 6, p. 733–745, 2001.

GENDA, K. On-demand network bandwidth reservation combining machine learning and linear programming. In: IEEE. **2021 17th International Conference on Network and Service Management (CNSM)**. [S.l.], 2021. p. 330–334.

GHALWASH, H.; HUANG, C.-H. A qos framework for sdn-based networks. In: **2018 IEEE 4th International Conference on Collaboration and Internet Computing (CIC)**. [S.l.: s.n.], 2018. p. 98–105.

GOOGLE. **IPv6 Adoption**. 2024. Google. Disponível em: <<https://www.google.com/intl/en/ipv6/statistics.html#tab=ipv6-adoption>>. Acesso em: 10.6.2024.

GOOGLE. **How much bandwidth does Skype need?** 2025. Disponível em: <<https://support.google.com/youtube/answer/78358?hl=en>>.

GREENSTEIN, S. The basic economics of internet infrastructure. **Journal of Economic Perspectives**, v. 34, n. 2, p. 192–214, May 2020. Disponível em: <<https://www.aeaweb.org/articles?id=10.1257/jep.34.2.192>>.

HAXHIBEQIRI, J. et al. In-band network monitoring technique to support sdn-based wireless networks. **IEEE Transactions on Network and Service Management**, v. 18, n. 1, p. 627–641, 2021.

HENTHORN-IWANE, A. **THE DIFFERENCE BETWEEN TIER 1 AND TIER 2 ISPS**. 2024. Inter.link. Disponível em: <<https://inter.link/blog/the-difference-between-tier-1-and-tier-2-isps>>. Acesso em: 10.6.2024.

HYPERLEDGER. **Introduction to Hyperledger Business Blockchain Design Philosophy and Consensus**. 2017. Hyperledger. Disponível em: <https://8112310.fs1.hubspotusercontent-na1.net/hubfs/8112310/Hyperledger/Offers/Hyperledger_Arch_WG_Paper_1_Consensus.pdf>. Acesso em: 10.6.2024.

JANEVSKI, T.; JANKOVIC, M.; MARKUS, S. **Quality of service regulation manual**. [S.l.], 2017.

JOSHI, K. D.; KATAOKA, K. Prime-q: Privacy aware end-to-end qos framework in multi-domain sdn. In: IEEE. **2019 IEEE Conference on Network Softwarization (NetSoft)**. [S.l.], 2019. p. 169–177.

JR, N. J. M.; PARPINELLI, R. S.; FIORESE, A. Classificação de tráfego para gerenciamento de largura de banda em redes definidas por software. In: SBC. **Anais do XX Encontro Nacional de Inteligência Artificial e Computacional**. [S.l.], 2023. p. 1022–1036.

JR., N. M.; PARPINELLI, R.; FIORESE, A. Classificação de tráfego para gerenciamento de largura de banda em redes definidas por software. In: **Anais do XX Encontro Nacional de Inteligência Artificial e Computacional**. Porto Alegre, RS, Brasil: SBC, 2023. p. 1022–1036. ISSN 2763-9061. Disponível em: <<https://sol.sbc.org.br/index.php/eniac/article/view/25761>>.

JÚNIOR, N. J. M.; FIORESE, A. Flowpri-sdn: A framework for bandwidth management for priority data flows applied to a smart city scenario. In: SPRINGER. **International Conference on Advanced Information Networking and Applications**. [S.l.], 2023. p. 346–357.

KARAKUS, M.; DURRESI, A. Quality of service (qos) in software defined networking (sdn): A survey. **Journal of Network and Computer Applications**, v. 80, p. 200 – 218, 2017. ISSN 1084-8045. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S1084804516303186>>.

KARAKUS, M.; GULER, E.; ULUDAG, S. Qoschain: Provisioning inter-as qos in software-defined networks with blockchain. **IEEE Transactions on Network and Service Management**, v. 18, n. 2, p. 1706–1717, June 2021. ISSN 1932-4537.

KERRISK, M. **ip-netns(8) — Linux manual page**. 2013. Disponível em: <<https://man7.org/linux/man-pages/man8/ip-netns.8.html>>.

KHAN, A. A. et al. Qos-ledger: Smart contracts and metaheuristic for secure quality-of-service and cost-efficient scheduling of medical-data processing. **Electronics**, MDPI, v. 10, n. 24, p. 3083, 2021.

KHANVILKAR, S. et al. **Multimedia networks and communication**. [S.l.]: Citeseer, 2004. 431–432 p.

Krainyk, Y.; Dvornik, O.; Krainyk, O. Software-defined network application-aware controller for internet-of-things. In: **2019 3rd International Conference on Advanced Information and Communications Technologies (AICT)**. [S.l.: s.n.], 2019. p. 165–169.

LACNIC. **LACNIC POLICY MANUAL**. 2023. LACNIC. Disponível em: <<https://www.lacnic.net/innovaportal/file/680/1/manual-politicas-en-2-19.pdf>>. Acesso em: 10.6.2024.

METZ, C. Interconnecting isp networks. **IEEE Internet Computing**, v. 5, n. 2, p. 74–80, 2001.

MICROSOFT. **How much bandwidth does Skype need?** 2025. Disponível em: <<https://support.microsoft.com/en-us/skype/how-much-bandwidth-does-skype-need-ad0fa9d7-c6ce-44ed-a3cd-5ea982df6e2a>>.

MOHANTA, B. K. et al. **Blockchain technology: A survey on applications and security privacy Challenges**. [S.l.]: Elsevier B.V., 2019.

MOORE, A.; ZUEV, D.; CROGAN, M. **Discriminators for use in flow-based classification**. [S.l.], 2013.

NIC.BR. **Fascículos sobre a Infraestrutura da Internet: Endereços IP e ASN - Alocação para Provedores Internet (Versão 2)**. 2022. Nic.br. Disponível em: <<https://www.nic.br/publicacao/fasciculos-sobre-a-infraestrutura-da-internet-enderecos-ip-e-asns-alocacao-para-provedores-int>>. Acesso em: 10.6.2024.

OLIVEIRA, A. T. et al. Sdn-based architecture for providing quality of service to high-performance distributed applications. **International Journal of Network Management**, Wiley Online Library, v. 31, n. 5, p. e2078, 2021.

ONF. **OpenFlow Switch Specification Version 1.3**. [S.l.], 2012.

ONF. **Software-Defined Networking: The New Norm for Networks**. [S.l.], 2012. Disponível em: <<https://pdfs.semanticscholar.org/a3f6/9f6181a0b4d481073a21eafbccc434a800db6.pdf>>.

(ONF), O. N. F. **Software-Defined Networking: The New Norm for Networks**. [S.l.], 2012.

OSTERMANN, S. **tcptrace**. 2003. Disponível em: <<http://www.tcptrace.org>>.

Paliwal, M.; Shrimankar, D.; Tembhurne, O. Controllers in sdn: A review report. **IEEE Access**, v. 6, p. 36256–36270, 2018.

PATIL, C. **APPLICATION LAYER PROTOCOLS**. 2023. Medium. Disponível em: <<https://medium.com/@chaituapatil3/application-layer-protocols-62ded821264e>>. Acesso em: 10.6.2024.

PFAFF, B.; PETTIT, J.; TOURRILHES, J. **ovs-fields–protocol header fields in Open-Flow and Open vSwitch**. 2021.

PHOKEER, A. **Interdomain routing security: Motivation and challenges of RPKI**. 76 p. Tese (Doutorado) — Royal Holloway, University of London, 08 2013.

PODILI, P.; KATAOKA, K. Traqr: Trust aware end-to-end qos routing in multi-domain sdn using blockchain. **Journal of Network and Computer Applications**, v. 182, p. 103055, 2021. ISSN 1084-8045. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1084804521000795>>.

RAHMAN, A. et al. Smartblock-sdn: An optimized blockchain-sdn framework for resource management in iot. **IEEE Access**, v. 9, p. 28361–, 02 2021.

RAIKAR, M. M. et al. Data traffic classification in software defined networks (sdn) using supervised-learning. **Procedia Computer Science**, v. 171, p. 2750–2759, 2020. ISSN 1877-0509. Third International Conference on Computing and Network Communications (CoCoNet'19). Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1877050920312928>>.

REIS, E. A. **PTTMetro, Interconexão de Sistemas Autônomos (AS)**. 2010. Cep-tro.br. Disponível em: <<http://old.ix.br/doc/nic.br-ceptro.br-pttmetro.apresentacao.pdf>>. Acesso em: 10.6.2024.

ROSEN, R. Namespaces and cgroups, the basis of linux containers. **Seville, Spain, Feb**, 2016.

SANTOS, F. U. dos. **O que são Jumbo Frames, para que servem, quando usar e quando não usar?** 2020. Monitoriz. Disponível em: <<https://skymonitor.com/pt/o-que-sao-jumbo-frames>>. Acesso em: 10.6.2024.

Selvaraj, P.; Nagarajan, V. Migration from conventional networking to software defined networking. In: **2017 International Conference on IoT and Application (ICIOT)**. [S.l.: s.n.], 2017. p. 1–7.

SERAG, R. H. et al. **Machine-Learning-Based Traffic Classification in Software-Defined Networks**. [S.l.]: Multidisciplinary Digital Publishing Institute (MDPI), 2024.

SHOKOUHYAR, S. et al. Improving internet service providers (isp) competitiveness: Isp's perception regarding customer satisfaction. **International Journal of Business and Systems Research**, v. 15, p. 1, 01 2021.

SHORE, M.; PIGNATARO, C. **An Acceptable Use Policy for New ICMP Types and Codes**. [S.l.], 2014.

SOBYTE. **Open vSwitch**. [S.l.], 2022. Disponível em: <<https://www.sobyte.net/post/2022-04/open-vswitch/>>.

TEAM, R. project. **RYU SDN Framework, Release 1.0**. [S.l.], 2014.

THAZIN, N.; NWE, K. M.; ISHIBASHI, Y. End-to-end dynamic bandwidth resource allocation based on qos demand in sdn. In: IEEE. **2019 25th Asia-Pacific Conference on Communications (APCC)**. [S.l.], 2019. p. 244–249.

TORRES, E. et al. A SDN/OpenFlow Framework for Dynamic Resource Allocation based on Bandwidth Allocation Model. **IEEE Latin America Transactions**, v. 18, n. 5, p. 853–860, abr. 2020. Disponível em: <<https://doi.org/10.5281/zenodo.3766178>>.

UNB. **VPN-nonVPN dataset (ISCXVPN2016)**. 2016. <https://www.unb.ca/cic/datasets/vpn.html>.

WANG, Z. et al. Internet multimedia traffic classification from qos perspective using semi-supervised dictionary learning models. **China Communications**, IEEE, v. 14, n. 10, p. 202–218, 2017.

XIAO, X. et al. A practical approach for providing qos in the internet backbone. **IEEE Communications Magazine**, IEEE, v. 40, n. 12, p. 56–62, 2002.

YE, N. Qos-centric stateful resource management in information systems. **Information Systems Frontiers**, Springer, v. 4, p. 149–160, 2002.

YOO, Y. et al. Teavisor: network hypervisor for bandwidth isolation in sdn-nv. **IEEE Transactions on Cloud Computing**, IEEE, v. 11, n. 3, p. 2739–2755, 2022.

APPENDIX A – FLOWPRI-SDN 2.0 SOURCE CODE

The implementation codes for FLOWPRI-SDN 2.0, along with the topologies and scripts created for this work, are available under the GPLv3 license at:

<https://github.com/NiltonMocelin/FLOWPRI-SDN2>

Alongside the codes, there are tutorials and explanations of design decisions, as well as solutions applied to each encountered problem along the way.

There are many dependencies to be installed before running the FLOWPRI-SDN 2.0. To install them, run the script *requirements.sh*.

The experiments can be replicated using the scripts to build and destroy the topology. These scripts are located in the folder *experimentos – NETNS/*. There is a Python notebook with more instructions also in this folder.