UNIVERSIDAD DE CONCEPCIÓN
FACULTAD DE INGENIERÍA
DOCTORADO EN CIENCIAS DE LA INGENIERÍA CON MENCIÓN EN INGENIERÍA ELÉCTRICA

# Explicit feedback for congestion control in Science DMZ cyberinfrastructures based on programable data-plane switches

Profesores supervisores: PhD. Miguel Figueroa, PhD. Jorge Pezoa (Q.E.P.D.)
Departamento de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de Concepción
Co-supervisor: PhD. Jorge Crichigno Benitez
College of Engineering and Computing
University of South Carolina

**Tesis para optar al grado académico de Doctor en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica**

**CHRISTIAN FERNANDO VEGA CAICEDO**

**CONCEPCIÓN - CHILE**

**2023**

# Prefacio

Esta tesis es presentada como parte de los requisitos para optar al grado académico de Doctor en Ciencias de la Ingeniería con mención en Ingeniería Eléctrica, de la Universidad de Concepción, Chile, y no ha sido presentada previamente para la obtención de otro título en esta Universidad u otras. La misma contiene los resultados obtenidos en investigaciones llevadas a cabo en el Departamento de Ingeniería Eléctrica, durante el período comprendido entre el año 2017 y 2023, bajo la dirección del Doctor Jorge Pezoa Núñez (Q.E.P.D) y posteriormente del Doctor Miguel Figueroa Toro. El trabajo contó con la co-supervisión del Doctor Jorge Chrichigno de la Universidad de Carolina del Sur.

**Chrsitian Vega Caicedo**
*christianvega@udec.cl*
*Departamento de Ingeniería Eléctrica*
*Facultad de Ingeniería*
*Universidad de Concepción*
*Concepción, 2023*

*A mi madre, mis hermanas y hermanos, sobrinos y sobrina...*
*A las personas que están y las que ya no están y de alguna forma*
*me acompañaron en este viaje de tres países e incontables tazas de café....*

# Agradecimientos

Esta tesis no hubiese sido posible sin el apoyo de:

Dr. Jorge Pezoa (Q.E.P.D.), profesor supervisor de la tesis, por sus grandes cualidades académicas y humanas. Resalto su capacidad de gestión y disposición de apoyo en todo momento, incluso en aquellos donde todo parecía perdido.

Dr. Miguel Figueroa, profesor supervisor quién como director del Doctorado y posteriormente como supervisor apoyo incondicionalmente mi anhelo de ser doctor.

Dr. Jorge Crichigno, quien puso a su disposición su el equipo técnico y humano del laboratorio de Ciberinfraestructura de la Universidad de Carolina del Sur, para la realización del componente experimental de la tesis. Su hospitalidad en Estados Unidos formará parte de mis mejores recuerdos.

Los profesores del Departamento de Ingeniería Eléctrica, Sergio Sobarzo, Sebastián Godoy, Daniel Sbarbaro, Pamela Guevara, Rosa Figueroa, Mario Medina, José Espinoza y Juan Tapia por brindarme una formación de alta calidad en sus especialidades y permitirme apoyar algunos de sus cursos como ayudante.

Los compañeros más cercanos del laboratorio de transmisión y el doctorado por todo su apoyo y los gratos momentos compartidos: Yasmany, Felipe, Andrés, Bárbaro, José Felix, Yaime, Silvia y Christopher. Ustedes hicieron de la UdeC mi segundo hogar.

A los colegas del CI-Lab de la Universidad de Carolina del Sur: José, Elie, Ali Mazlum y Ali Alsabeth por todo su apoyo profesional y amistad en los meses que estuve de visita.

El equipo de apoyo administrativo del Departamento de Ingeniería Eléctrica en especial a Marcela Hernández e Inés Gonzales.

La Agencia Nacional de Investigación y Desarrollo (ANID) que soportó financieramente mis estudios con la Beca Doctorado Nacional Folio: 2018-21180418.

La Unviersidad CESMAG, que me permtió acceder a la comisión de estudios doctorado mediante el convenio de complementación académica 004-17.

# Contents

# List of Figures

# List of tables

# List of Acronyms

**ACK** Acknowledgement

**ALU** Arithmetic Logic Unit

**ANN** Artificial Neural Network

**ASIC** Application-Specific Integrated Circuit

**ATTR** Average Two-Terminal Reliability

**API** Application Programming Interface

**AQM** Active Queue Management

**BBR** Bottleneck Bandwidth and Round-trip

**BBR2** Bottleneck Bandwidth and Round-trip version 2

**BDP** Bandwidth Delay Product

**CCA** Congestion Control Algorithm

**CDF** Cumulative Distribution Function

**CNID** National Council of Innovation for Development

**CODEL** Controlled Delay

**CORFO** Corporation for the Production Promotion

**CWND** Congestion Window

**DFTCP** Delay Friendly TCP

**DMZ** Demilitarized Zone

**DPI** Deep Packet Inspection

**DTN** Data Transfer Node

**D-TCP** Dynamic TCP

**ECN** Explicit Congestion Notification

**ESnet** Energy Science Network

**JSON** JavaScript Object Notation

**FCT** Flow-Completion Time

**FPGA** Field Programmable Gate Array

**FS** Flow Size

**GA** Genetic Algorithm

**GNB** Gaussian Naive Bayes

**GSLRP** Geo-Safe Links Re-utilization Problem

**ICT** Information and Communication Technology

**IETF** Internet Engineering Task Force

**IP** Internet Protocol

**IoT** Internet of Things

**IDT** Inter-departure Time

**IDS** Intrusion Detection Systems

**IAE** Integral Absolute Error

**IT** Information Technology

**KNN** K-Nearest Neighbors

**LS** Least Squares

**LPR** Lost Packet Rate

**LPRH** Lost Packet Rate on Handover

**ML** Machine Learning

**MLC** Machine Learning Controller

**MLPC** Multi-Layer Perceptron Classifier

**MPTCP** Multipath TCP

**MRAE** Mean Relative Absolute Error

**NAT** Network Address Translation

**ONOS** Open Network Operating System

**OVS** Open vSwitch

**P4** Programming Protocol-Independent Packet Processors

**PDP** Programmable Data Plane

**PIE** Proportional integral Controller Enhanced

**PISA** Protocol-Independent Switch Architecture

**PSNR** Peak Signal-to-Noise Ratio

**QoE** Quality of Experience

**QoS** Quality of Service

**RBF** Radial Basis Function

**RED** Random Early Detection

**REUNA** National University Network

**RDE** Research, Development, and Entrepreneurship

**RDI** Research, Development, and Innovation

**RL** Reinforcement Learning

**RTT** Round Trip Time

**SBD** Scientific Big Data

**SDN** Software-Defined Networking

**SNMP** Simple Network Management Protocol

**SVM** Support Vector Machine

**SVR** Support Vector Regression

**TCP** Transmission Control Protocol

**TBF** Token Bucket Filter

**UDP** User Datagram Protocol

**TB** Tera bytes

**UDEC** Universidad de Concepción

**UDP** User Datagram Protocol

**VM** Virtual Machine

**VoIP** Voice over Internet Protocol

**WAN** Wide Area Network

# Abstract

A massive amount of scientific data is generated daily in different areas of knowledge. The challenges of research are increasingly sophisticated and demand a higher amount of information to make decisions, and also technological tools are increasingly more available for researchers. In many cases, this information, in addition to being collected and processed, must be transmitted among different research centers located over considerable distances. Internet and other dedicated cyberinfrastructures are needed to achieve this goal. Transferring scientific information between remote sites involves challenges in performance, security, coexistence, and resource allocation. Therefore, Energy Science Network (ESnet) proposed the concept of Science DMZ that provides design patterns for an optimized network environment to exchange scientific data.

Due to their nature, the Scientific Big Data (SBD) flows are connection-oriented, demanding high and constant throughput under low delay and jitter to reach reasonable transmission times. This imposes fundamental challenges to network design, especially congestion control mechanisms, buffer estimation, and anomaly detection, which are tackled in this thesis. This dissertation addresses two disruptive technologies to respond to these challenges: data-driven models and data-plane programmable devices.

We evaluated our solutions through production networks and testbeds using a hardware-based data plane and routing devices. Results showed that the proposed solutions could effectively improve the networks' performance and effectively adapt the Science DMZ design to non-dedicated networks.

# Resumen

Diariamente, se genera una gran cantidad de datos científicos en diferentes áreas del conocimiento. Los desafíos de investigación son cada vez más sofisticados y demandan una mayor cantidad de información para tomar decisiones, y además, las herramientas tecnológicas están cada vez más al alcance de los investigadores. En muchos casos, esta información debe ser recopilada y procesada, y transmitida entre diferentes centros de investigación ubicados a considerable distancia. Se recurre Internet y otras ciberinfraestructuras dedicadas para lograr este objetivo. La transferencia de información científica entre sitios remotos implica desafíos en el rendimiento, la seguridad, la coexistencia y la asignación de recursos. Por lo tanto, ESnet propuso el concepto de Science DMZ que proporciona patrones de diseño para un entorno de red optimizado para intercambiar datos científicos.

Debido a su naturaleza, los flujos de datos científicos masivos son orientados a la conexión, demandando un alto y constante tasa de transmisión, con baja latencia y variabilidad deseables, para alcanzar tiempos de transmisión razonables. Esto impone desafíos fundamentales al diseño de la red, especialmente en los mecanismos de control de congestión, estimación de los buffers y detección de anomalías, temas objeto de estudio de la presente tesis. Esta disertación aborda dos tecnologías disruptivas para responder a estos desafíos: modelos impulsados por datos y dispositivos programables en el plano de datos.

Las soluciones fueron evaluadas empleando redes de producción y testbeds utilizando dispositivos reales de enrutamiento y procesamiento en el plano de datos. Los resultados mostraron que las soluciones desarrolladas podrían mejorar efectivamente el rendimiento de las redes académicas y adaptar efectivamente los patrones de diseño de Science DMZ a redes no dedicadas.

# 1 Introduction

Current scientific applications, such as collaborative research experiments, astronomical observations, weather prediction, and hyperspectral imaging in medicine or biology, may quickly generate Tera bytes (TB) of data daily. For instance: (i) the Belle 2 High Energy Physics experiment is expecting to collect at least 250 PB of raw data in its first five years of operation [1]; (ii) during the second run, the CERN computing center has saved up to 10 GB of data per second, this information was transmitted to globally distributed computing centers by using the grid computing paradigm [2], and (iii) instruments used in the earth remote sensing could generate hyperspectral image datasets in terms of GB [3]. The Chilean scientific community also faces this overflowing growth in generating scientific data. In the north of Chile, the ALMA radio observatory generates approximately 600 GB daily, the AURA optical observatory produces 1 TB per night, and the Large Synoptic Survey Telescope, currently under construction, is expected to generate 28 TB per day. SBD is usually transferred to processing and storage sites, which may be located from hundreds of meters to thousands of kilometers. A specialized computer network device, Data Transfer Nodes (DTNs), is used to facilitate high-speed transferring. These systems are typically PC-based Linux servers designed expressly for wide-area data transfer and constructed with high-quality components. The DTN also has high-speed access to the storage system and runs software tailored for SBD transmissions. In the case of ALMA and AURA, these observatories have 2 Gbps dedicated network links from their sites in the north of Chile to facilities in Santiago. Subsequently, SBD must be exchanged over an international long-haul to Europe and the USA [4]. However, even though such links are dedicated, they are not exclusively used for transferring SBD since Internet data, emails, and phone calls are also exchanged over them.

Regardless of the distance separating data sources and processing sites, SBD is transferred over non-dedicated networks, i.e., SBD is exchanged over general-

purpose networks, and massive scientific data must share limited bandwidth and network resources with regular network traffic, such as Internet browsing, Voice-over-IP, audio streaming, and video streaming. While SBD calls for low packet loss rates and sustained high throughput [5], the above-mentioned data classes demand diverse Quality-of-Service Quality of Service (QoS) [6]. An unwanted behavior occurs when the output rate of a terminal is increased arbitrarily. In this case, DTN, the queues of the routers overflow, and a congestion collapse can be generated [7]. Despite the availability of high-speed, low-latency links, adding more bandwidth is not a solution because shared networks do not meet the requirements for SBD transmission on schedule, and transfer protocols also impose further restrictions that dramatically reduce throughput. Moreover, when SBD is exchanged over high-speed, high-latency links, transfer times of massive data may take several days or weeks. The overall result for scientists and network architects is frustrating: SBD is exchanged using flash drives and hard drives even when nodes are interconnected and separated by a few blocks.

One of the main concerns that condition the performance of stream transmissions in non-dedicated networks is the buffer size of both the transmitting nodes and the routing devices. Bufferbloat and packet rejection are problems associated with improper sizing or estimation of buffer sizes. Therefore solutions related to congestion control and suitable sizing and estimation of buffers are required to ensure adequate performance in shared cyberinfrastructures.

In addition, shared network infrastructures often use firewalls that perform a fine-grained inspection of the flows that cross the network, which can generate considerable performance degradation when transmitting flows. One of the premises in the Science-DMZ design pattern is to have friction-free paths for this flow type. Therefore, it is required to propose new traffic inspection schemes that have a reduced impact on overall performance.

Recently, programmable devices in the data plane of the Software-Defined Networking (SDN) layer model, have attracted the attention of researchers, architects, and network administrators because they allow line-rate processing, facilitating accurate measurements of network status. This technology could therefore support novel traffic control schemes.

Moreover, the last decade has seen exponential growth in the capabilities of machine learning or data-driven algorithms to generate estimates in a wide variety of fields. Data networks are no stranger to this trend, and there are many initiatives around improving network performance regarding security, traffic control, and management.

Considering the above, the following research questions arise: Is it feasible to adapt the Science DMZ design pattern to non-dedicated networks? Could data-driven models support buffer size estimation and anomaly detection in shared networks? Using programmable devices in the data plane in conjunction with

suitable Machine Learning algorithms, could these be the key technologies to enable the transfers of SBD flows over non-dedicated networks?

## 1.1 Hypothesis and research questions

The hypothesis that was tested during this doctoral thesis is:

The design of congestion control protocols based on explicit feedback mechanisms that are implemented using P4 Programmable Data Plane (PDP) switches would improve the performance of a Science DMZ cyber-infrastructure in terms of flow completion time, utilization, and fairness while collecting and processing reduced network state information.

The research questions driving this thesis are:

i) How to represent the main components of a high-performance SBD cyber-infrastructure under the Science DMZ design pattern?

ii) How to emulate packet loss and delay scenarios to determine the behavior of conventional implicit feedback mechanisms?

iii) How to integrate PDP Switches into a Science DMZ infrastructure?

iv) What are explicit feedback mechanisms approaches compatible with current protocols feasible to implement in PDP switches?

v) Which Machine Learning approach allows modeling more effectively source traffic and explicit feedback mechanisms in a Science DMZ cyber-infrastructure?

vi) How to validate the proposed explicit feedback mechanisms?

vii) How to program feedback mechanisms in PDP switches?

viii) How to reach a suitable coexistence between short flows and SBD over non-dedicated Networks?

## 1.2 Objectives

### General objective:

Develop explicit feedback congestion control mechanisms from PDP switches to improve the performance of a Science DMZ cyber-infrastructure.

### Specific objectives:

i) Implement a P4-based testbed for Science DMZ cyber-infrastructure that allows the exchange of Scientific Big Data and the implementation of performance measurements in the data plane.

ii) Model network resources and protocols to abstract the Science DMZ cyber-infrastructure components and optimize their behavior

iii) Use Machine Learning controllers to improve the SBD flow transmissions in a Science DMZ

iv) Define novel explicit feedback mechanisms for congestion control based on PDP switches capabilities.

v) Evaluate the proposed mechanisms' performance regarding target rate, occupancy, and flow completion time.

## 1.3 Author contributions.

The primary author's contributions to the state-of-the-art of explicit traffic control in Science DMZ using PDP devices are the formulation of a model that abstracts the network state of a Science DMZ from passive data plane measurements, the development of a Machine Learning Controller (MLC) that uses the network state to control the flows in the network in order to improve the SBD transmissions, and the implementation of novel rate control strategies in a Science DMZ testbed. In addition, problems related to buffer size estimation and anomaly detection, which are part of the Science DMZ design, were addressed during the Ph.D. studies. Finally, considering that P4 is an innovative technology in the evolution of SDN and could be adopted within the curricula in engineering programs, a framework for teaching and research in that area of knowledge was proposed.

### 1.3.1 Network state abstraction from data plane measurements.

Considering the network as a dynamic system that can be described through a set of variables, we leverage the data plane capabilities to collect variables at the network bottleneck: rate, Round Trip Time (RTT), queuing delay, and the number of active flows. The above implied the development of codes in P4 that would allow the collection of the necessary information and reporting it timely to the control plane.

### 1.3.2 MLC to improve SBD flows over non-dedicated networks

Two data-driven models to control SBD flows over a non-dedicate network were built from network state traces collected through extensive experiments on a real testbed. One of the models was trained when the DTN operates with Cubic as the congestion control algorithm, while another model was developed when using Bottleneck Bandwidth and Round-trip version 2 (BBR2). On the one hand, Cubic is the default Congestion Control Algorithm (CCA) for several operating systems because it indirectly estimates network congestion and reacts to changes in network conditions. Its name is due to the cubic function used to increase the Transmission Control Protocol (TCP) window size. Among the benefits of Cubic are high scalability and fairness in networks with homogeneous conditions. On the other hand, BBR2 is based on bandwidth and RTT estimation, offering higher throughput and optimized latency. For the above reason, it is a suitable CCA for DTN on Science DMZ networks. The proposed models use the foundations of direct-inverse control supported by Artificial Neural Networks (ANNs). The MLC takes the current and past network states to compute the output rate of the campus network to control the congestion produced by short flows in the non-dedicated network.

### 1.3.3 Coarse Estimation of Bottleneck Router's Buffer Size on heterogeneous networks

An estimator of the buffer size at the bottleneck in a non-dedicated network was developed. The model leverages end-to-end measurements to predict an operating regime of the buffer size at the bottleneck link. Exploratory analysis of the patterns showed that these classes or regimes are not linearly separable. Therefore Machine Learning models were trained in order to discriminate between these regimes. The classifier also works when several CCA coexist in the network, a common aspect in non-dedicated networks.

### 1.3.4 Entropy Characterization of NATed Campus Networks

We proposed a framework for characterizing the entropy of traffic traversing a Campus production network based on established flows instead of traditional packet-counting approaches. In addition, the developed work considers that the Campus Network uses the Network Address Translation (NAT) service to allow hosts to access the Internet. This aspect differs the proposal from the existing ones. The solution generates a negligible impact on the network's operation as

opposed to conventional firewalls. The entropy characterization allowed the establishment of normal and anomaly operation ranges.

### 1.3.5 A Novel Framework for SDN Teaching and Research

SDN is a networking paradigm that separates the network control and data planes, allowing for centralized, software-based management and configuration of the network. This approach provides greater flexibility and programmability to the network, enabling more efficient use of network resources and easier management of network policies. PDP switches are a key enabler of SDN. They provide the necessary flexibility and programmability, allowing precise network telemetry and offloading functions traditionally belonging to the control plane, offering dynamic network control and management. SDN can provide several ways to support a Science DMZ, including network segmentation, traffic prioritization, dynamic resource allocation, security, and network monitoring and analytics.

As a contribution to the Electrical Engineering Department of the University of Concepción, and particularly to the telecommunications engineering program, a framework for teaching and research in SDN was structured. The key idea is to leverage the close relationship between SDN, real-world ICT problems, and innovative services development. Thus, the proposed framework integrates actors, stakeholders, phases, components, and interrelationships in an applied research environment. The above ensures that the contributions of this Ph.D. thesis and other related work are fed back into the curriculum, thus facilitating the development of new initiatives in the field.

### 1.3.6 Journal papers and conference presentations.

The following journal papers and conference presentations were obtained as a result of this thesis, and they allowed the dissemination of this research work:

- Vega C. Kfoury, E. Gomez, J., Pezoa, J. Figueroa, M., and Crichigno, J. (2023). Machine Learning Controller for Data Rate Management in Science DMZ Networks (under review). Computer Networks.

- Vega, C., Pezoa, J. E., Kfoury, E. F., and Crichigno, J. (2021). Coarse Estimation of Bottleneck Router's Buffer Size for Heterogeneous TCP Sources. In 2021 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1-6). IEEE.

- Vega, C., Prieto, Y., Pezoa, J. E., Sobarzo, S. K., & Ghani, N. (2019). A Novel framework for SDN teaching and research: A Chilean University case study. IEEE Communications Magazine, 57(11), 67-73.

- Prieto, Y., Vega, C., Pezoa, J. E., and Crichigno, J. (2019). Shared-risk-aware Design for Survivable Migration in SDN Environments. In 2019 16th IEEE Annual Consumer Communications and Networking Conference (CCNC) (pp. 1-6). IEEE.
- Crichigno, J., Kfoury, E., Bou-Harb, E., Ghani, N., Prieto, Y., Vega, C., and Torres, D. (2019). A flow-based entropy characterization of a NATed network and its application on intrusion detection. In ICC 2019-2019 IEEE International Conference on Communications (ICC) (pp. 1-7). IEEE.

## 1.4 Thesis Manuscript Structure

The thesis manuscript presented here is organized as follows. Chapter 2 presents the related work on the topics addressed by this study. Also, it frames the role of data-plane devices and Machine Learning (ML) techniques to improve the performance of SBD over non-dedicated networks. Chapter 4 tackles the bottleneck router's buffer size coarse estimation problem using Machine Learning models. In Chapter 3 describes and evaluates the Machine Learning Rate Controller to outperform SBD transmissions over non-dedicated networks. Chapter 5, a flow-based entropy characterization of academic networks, is proposed to detect possible traffic anomalies. Chapter 6 presents a teaching and research framework to integrate the findings of SDN related projects to the academy. Finally, Chapter 7, presents the conclusions and main contributions of the present study, as well as future work based on the findings and results.

# 2 State-of-the-art and motivations

The present chapter presents the state-of-the-art of Scientific Big Data (SBD) transmissions in non-dedicated networks. Firstly we address the concept of Science DMZ and its relevance for data-intensive transmissions for scientific purposes. Secondly, we review the Programmable Data Plane (PDP) approach and its applications. Thirdly we discuss the end-to-end and network-assisted rate control strategies and review the role of Machine Learning (ML) in improving traffic control. Fourthly, we present the common issues of large data transfers over non-dedicated networks and the engineering solutions to deal with them. Finally, we conclude the state-of-the-art with some remarks and motivations.

## 2.1 Science DMZ

The term Science DMZ was first introduced in the Energy Science Network (ESnet) by Dart et al. as a design pattern that is suitable for optimizing the interactions between Wide Area Networks (WANs), campus networks, and computing systems [8]. The so-called "Science DMZ paper" also addresses use cases where the Science DMZ pattern has been set into practice at the University of Colorado, Pennsylvania State University, Virginia Tech Transportation Institute, National Oceanic and Atmospheric Administration, and the National Energy Research Scientific Computing Center. The authors also claim that there is a need to improve the transport protocols, the extension of virtual circuits, the adoption of high throughput standards, such as 100-Gigabit Ethernet, and the deployment of Software-Defined Networking (SDN) solutions to enhance the performance of the SBD exchange. Several institutions and academic collaboration networks worldwide [9, 10, 11] have joined this initiative, favoring the technological platform for exchanging large volumes of scientific information.

The main motivations for adopting the Science DMZ design pattern are as

follows:

- *Collaboration*: Many scientific experiments require collaboration between researchers in different parts of the world. The Science DMZ provides a platform for seamless collaboration, enabling researchers to share data and resources with colleagues regardless of their location, which can significantly enhance the efficiency and effectiveness of scientific research.

- *Optimized Network Performance*: The Science DMZ provides a dedicated network environment optimized for SBD transfers with minimal friction. Engineers focus on designing, configuring, and optimizing network infrastructure to ensure it can handle the high volumes of data generated by scientific experiments.

- *Scalability*: The Science DMZ architecture is designed to scale up or down as needed, allowing researchers to handle the increasing volume of data generated by modern scientific experiments. This scalability can help the network handle large data volumes and quickly adapt to changing research needs.

- *Security*: SBD transfers need to be secured to keep the integrity and confidentially of scientific traffic with the minimum overhead. Therefore novel approaches for access controls, monitoring, and auditing capabilities are needed.

Figure 2.1 depicts an academic network cyberinfrastructure. First, consider only the network outside the dotted frame that corresponds to a regular Campus Network where the resources are shared among all users and frictions generated by devices such as firewalls are present. Also, the Campus Networks support heterogeneous traffic in terms of delay and flow size; hence the overall performance can be highly affected. Now consider including the network segment of the dotted frame in the cyberinfrastructure, which shows the essential elements of the Science DMZ cyberinfrastructure, according to the comprehensive tutorial of [6]. In the architecture, Data Transfer Nodes (DTNs) are specialized Linux devices built and configured for generating and receiving SBD traffic, at high rates, from and to external labs. The proposed network topology also considers using a monitoring server to track the connections and automatically generate reports of end-to-end performance metrics. Besides, an offline security appliance, like an Intrusion Detection Systems (IDS), is used to protect the network in a timely fashion. A storage system, designed according to the requirements of the SBD application, is necessary to allocate the raw and processed data either acquired or generated, for instance, by the supercomputer.

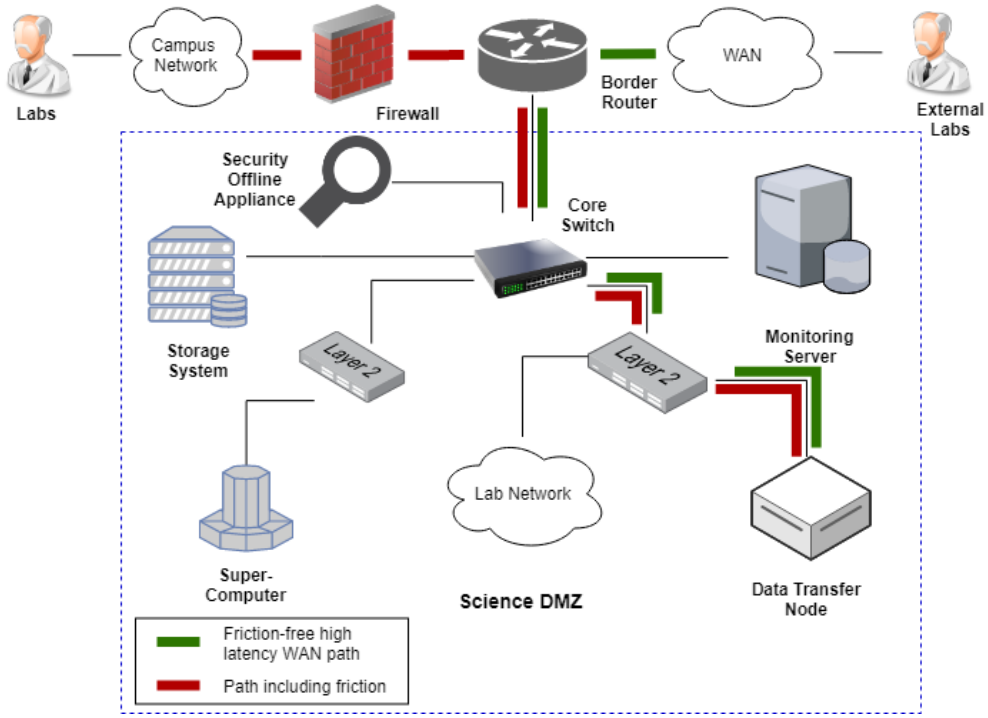



**Figure** 2.1: Science DMZ cyberinfrastructure components.

From the state-of-the-art in Science DMZ research, two issues have been identified where efforts have been focused: performance and security. Regarding performance, ESnet proposes a set of good practices for network device selection and DTN tuning. The use of pacing, parallel flows, and novel congestion control protocols such as Bottleneck Bandwidth and Round-trip (BBR) are recommended strategies to maximize the performance of SBD transmissions [12, 13]. Network architects also have exploited SDN technology to deploy virtual circuits, thereby dealing with SBD traffic in actual network implementations. The authors in [1] examined architectural models for leveraging OpenFlow switches and the SDN model within the science-networking context. Then, they presented designs for SC12 SCinet Research Sandbox. In Thailand, the National Research and Education Network was boosted by the adoption of SDN to control the operation on six research DMZ nodes [14]. The results showed that the throughput among these nodes increased from 100 Mbps to 900 Mbps. The AmoebaNet, an SDN-enabled service for SBD [15], was proposed to address three significant insights in Science DMZ: the last mile problem, the scalability problem, and the programmability problem. The solution considered traffic control based on Quality of Service (QoS) policies, obtaining acceptable performances in differentiated traffic. However, it required the addition of functions specified in the SDN controller

and modifications in the DTNs.

Conventional firewalls are unsuitable for Science DMZ due to the performance degradation they present at high transmission rates. One of the main concerns is the security in this particular deployment type. Several authors have addressed these problems by implementing non-invasive mechanisms for traffic analysis in Science DMZ. Several authors have addressed these problems by implementing non-invasive mechanisms for traffic analysis in Science DMZ. Insider attack detection system for Science DMZ is implemented in [16] by measuring the CPU load, disk usage, along with network activity. In [17], efficient traffic monitoring for Science DMZ is studied, avoiding the typical Deep Packet Inspection (DPI) mechanism and considering a lightweight side-channel based abnormality detection supported by a basic rule.

## 2.2 Programmable Data Plane Switches

Last decade's advances in the development of Application-Specific Integrated Circuits (ASICs) and Field Programmable Gate Arrays (FPGAs) showed that it is possible to achieve Tbps forwarding speeds with a set of packet processing capabilities [18, 19, 20]. These capabilities enabled the hardware designers to abstract flexible functions, such as parsing packets and matching header fields, thereby allowing SDN controller applications to leverage these capabilities through a common open interface in the data plane. A high-level programming language is required to take advantage of these abilities effectively. Besides, hypervisors use software switches to steer packets to and from Virtual Machines (VMs) in data centers. These software switches are typically based on a large body of code, which is challenging to manage and update [21]. The P4 open-source language emerged immediately as a feasible solution that tackles three main goals of yielding a data plane programmable network device: reconfigurability, protocol independence, and target independence. P4 is a language extensible for ASICs, FPGAs, and virtual-based switches, among other devices [22].

P4 uses Protocol-Independent Switch Architecture (PISA), which is a pipeline forwarding architecture for the programming data plane. Figure 4 depicts PISA, which is composed of three main programmable elements: (i) a parser that extracts packet headers based on a defined policy; (ii) a set of match-action stages organized in pipeline, which perform operations like packet ordering or filtering, form the headers using Arithmetic Logic Units (ALUs) with computational capabilities and stateful memories; and (iii) a deparser that reassembles the packets that the output ports will forward. The P4 language Ecosystem is overgrowing with a wide range of products, projects, and services that leverage P4 [23].

Figure 2.3 shows the workflow for the design and implementation of P4-based

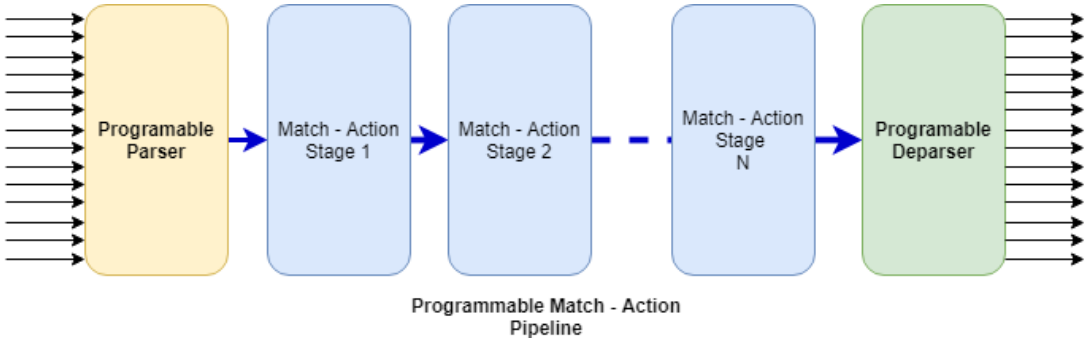



**Figure** 2.2: Protocol-Independent Switch Architecture

data planes. The first step is to check the architecture model of the target. The architecture here means the function blocks present in the target and the interfaces between them. To be aware of the architecture, let the programmer know the available resources and maximize them to develop a given task. Although architectures may vary between manufacturers, the Programming Protocol-Independent Packet Processors (P4) Consortium has made an effort to define a minimum set of generic architecture elements. This gave rise to the P4 Language Specification, whose current version is $P4_{16}$ [24]. Writing programs in P4 are mandatory to add the standard library called core.p4 and a library related to the target architecture.

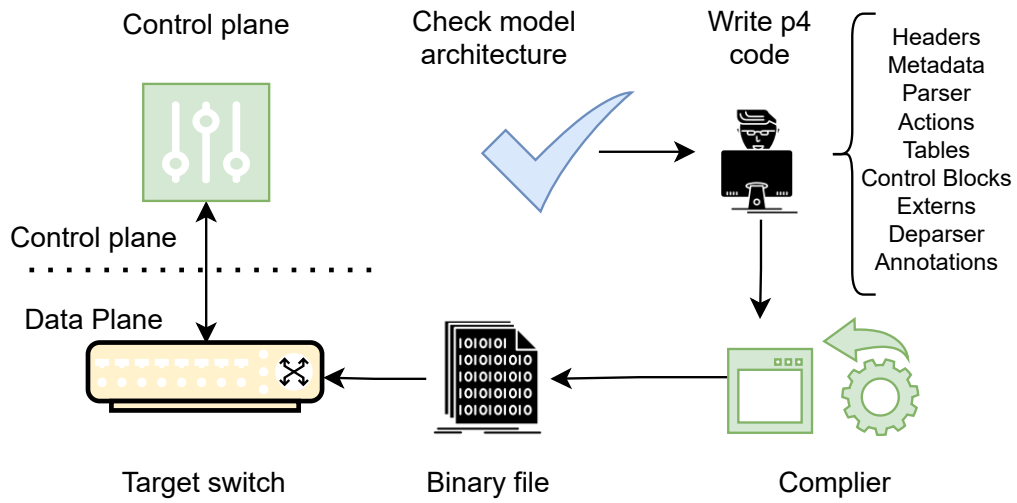



**Figure** 2.3: P4 Workflow

Writing the code is a matter for the programmer and involves the development of the following abstractions:

- *Header:* It is a grouping of variables that represents the data corresponding

to the header of a protocol present in the packet (e.g., IPV4).

- *Metadata:* It Refers to additional information about a packet not part of the packet header. A network device can add metadata as the packet passes through it. This information can be used to make forwarding decisions, apply policies, or perform other operations on the packet.

- *Parser:* This programmable block aims to extract the values of the packet header that will be processed by the data plane. The parser is typically the first stage of the packet processing pipeline in a P4 architecture, and it performs header extraction, field extraction, field validation, and header stacking. The programmer must model the state machine representing all combinations in the order of appearance of the packet headers and program the states and transitions.

- *Actions:* These are a set of sequences that can process arguments and return values according to a specific action programmed, such as: changing the egress port of the packet, changing the MAC address, dropping the packet, or sending the packet to the control plane for further processing.

- *Tables:* These abstractions are entities composed of entries on which the control plane establishes actions of the P4 program. In the tables, values established in the headers or metadata are mapped with one or more actions. Each table entry has a match key, such as a destination IP address and source port of the packet. For consistency, match keys must be unique and can only appear once in table entries. Depending on the target, the tables can be defined as additional properties such as expiration time, a maximum number of entries, or the default action. Tables in P4 are typically organized into a pipeline, where each table performs a specific function in the packet processing pipeline. The output of one table may be used as the input for the following table in the pipeline. The above allows for flexible and modular packet processing, where different pipelines can be defined for different network protocols or use cases

- *Control Blocks:* Due to PISA architecture's pipeline nature, the *control blocks* are limited, therefore in P4 *for* or *while* are not supported. In a P4 code, it can only include *if ... else*, *switch* and *hit ... miss* control blocks. The *hit..miss* statement defines two exclusive code blocks to execute depending on the table's search result. The architecture establishes the number and location of the control blocks.

- *Externs:* These are specific objects of each target that can be configured from the control plane. Functions for efficient hash or checksum calculation

are examples of externs used recurrently in P4 programs.

- *Deparser:* The deparsers are intended to perform the opposite function to the parser. These serialize the headers modified by the programmer on the payload of the packet. In this context, the payload corresponds to the set of bytes in the packet that the parser has not processed.
- *Annotations:* These placeholders are special directives with a high range of applications, such as: limiting the use of an action to a context, enabling or hiding elements to the control plane, setting limits for objects, and performing code verification [25].

The compilation process is carried out with the P4 source code ready, transforming it into an understandable language for the target platform. This target-specific code is loaded in the P4 switch. When the switch operates, it can interact with the control plane through Application Programming Interfaces (APIs). The default API implementation for communicating the control plane with the data plane is P4Runtime. It can be used to insert, update, and delete entries in P4 tables and manipulate P4 programming elements such as counters and meters.

According to a recent exhaustive survey [26], programmable switches are applied in In-band Network Telemetry, Network Performance, Middlebox Functions, Accelerated Computing, Internet of Things (IoT), and Cybersecurity. Although a large part of the developments related to P4 was carried out at an experimental level, the authors envision that this technology will be increasingly integrated into cyberinfrastructures.

## 2.3 Rate Control

Rate control in networks is a crucial area of study and continuously evolves with network technologies' growth. The goal of rate control is to manage network resources efficiently, ensure fair utilization, and provide a high-quality of service to the end-users. Rate control strategies have been developed into end-to-end and network-assisted categories. In the first one, the end devices autonomously regulate the output rate according to indirect measurements they can make on the state of the network. In contrast, network-assisted rate control is achieved by having a central entity, such as a controller, which monitors and regulates the data flows on the network.

### 2.3.1 Legacy end-to-end rate control

The default end-to-end rate control used on the Internet is based on the Transmission Control Protocol (TCP) [27] whose algorithms have been refined over time. One of

the most popular flavors of TCP Congestion Control Algorithm (CCA) is Cubic, which is the default CCA in Linux Systems. Cubic is a loss-based CCA; they rely on packet loss to detect congestion. However, when short flows compete with long flows, even a few of the latter, bandwidth starvation occurs [28], affecting overall network performance. In 2016 Google released BBR [29], a disruptive model-based CCA that estimates the bandwidth and Round Trip Time (RTT) to infer the Bandwidth Delay Product (BDP) and computes the host output rate. Authors in [30] conclude that BBR works well for a single long flow at a bottleneck. However, it presented coexistence issues with CUBIC, evidenced by the fairness index in tests performed in [31] and [32]. Subsequently, Bottleneck Bandwidth and Round-trip version 2 (BBR2) [33] was released to mitigate the drawbacks presented by the first version by using Explicit Congestion Notification (ECN) and packet-loss rate estimation. Tests conducted in [32] and [34] confirmed that BBR2 outperforms BBR in terms of fairness and packet loss.

### 2.3.2 Legacy network-assisted rate control

Legacy network-assisted rate control mechanisms rely on router's Active Queue Management (AQM) algorithms. One of the first AQM algorithms is Drop-Tail, in which the packets from all hosts are accepted until the queue size is reached. Floyd and Jacobson in [35] proposed Random Early Detection (RED), which performs probabilistic packet dropping to inform the hosts about network congestion. The higher the queue occupancy in the router, the higher the probability of dropping. In the test performed, RED performed better than Drop-Tail. However RED some drawbacks, including relying only on the queue occupancy, the lack of knowledge of the number of flows that share the bottleneck, complex parameter tuning, and not being suitable for small buffers [36, 37]. Controlled Delay (CODEL) was proposed in [38] focusing on router queuing delay. CODEL tracks the sojourn time and the time the packet takes between router ingress and egress. If the sojourn time exceeds a pre-defined threshold, the packet is discarded in the de-queue process [36]. Performance tests, such as those conducted in [39], showed that CODEL overcomes RED regarding queuing delay and link utilization. Despite the advantages mentioned above, CODEL has issues with the router's memory usage and scalability due to the additional computational load involved in the queuing delay computation. Afterward, CODEL, Proportional integral Controller Enhanced (PIE) [40] was proposed by Internet Engineering Task Force (IETF), with the premise of offering the best of RED and CODEL. PIE relies on control theory to estimate the dropping probability based on queuing latency computation. Tests performed in [41] demonstrated that PIE works better than CODEL in terms of queuing delay under heavy congestion.

### 2.3.3 Machine-Learning-based rate control

Due to the heterogeneous and complex traffic in today's networks, it is not feasible for a rate control mechanism to work properly in different scenarios. Therefore, in recent years, initiatives have emerged that involve learning as an element to be considered in the modeling and operation of rate control algorithms. There is a set of End-to-end performance-oriented, involving upgrades in network end devices. Remy [42], PCC [43], PCC Vivace [44], PCC Proteus [45], GCC [46], Copa [47], Indigo [48], and ZiXia [49] are based on the calculation of an objective function that considers measurements of the network state. The objective function parameters in the above algorithms are tuned using recurrent neural networks or reinforcement learning. These solutions have shown significant improvements over legacy rate control algorithms. Other learning-based solutions train ML models from datasets used supervised, unsupervised and Reinforcement Learning (RL) techniques. A comprehensive survey about the previous end-to-end rate control solutions based on ML is found in [50].

Although research focuses on end-to-end ML-based rate controllers, there is a set of solutions for the network-assisted approach as an enhancement of the AQM algorithms. Neural networks [51, 52], Q learning [53, 54, 55] and RL [56] are the preferred ML algorithms. The above network-assisted rate control mechanisms operate with limited functions restricted by vendor or software routers with performance impairments that make them impractical for high-performance applications [57].

### 2.3.4 Rate control supported by Programmable Data Planes

Last decade's advances in the development of ASICs show that it is possible to achieve terabit forwarding speeds with a set of packet processing capabilities. These capabilities enable the creation of network devices able to monitor and control network traffic at line-rate in the data plane. P4 emerged as a de-facto data plane programming language that tackles three main goals: reconfigurability, protocol independence, and target independence because it is extensible for ASIC, FPGA, and virtual-based switches, among others [58]. P4 has enabled the deployment of rate control enhancements from various approaches, including host-centric, enhanced feedback, traffic isolation, and fast rerouting [59]. Some solutions [60, 61, 62, 63, 64] require software modifications to the end-hots and would therefore be applicable when the network administrator has full control of the network devices. Legacy AQM algorithms have been adapted and implemented on P4 switches with certain limitations [65, 66, 67, 68, 69, 70]. P4 also enabled the fast development of custom AQM algorithms such as P4-SRPT [71], P4-ABC [72], SP-FIO [73], FDPA [74], P4QoS [75] and QoSTCP [76]. The above

schemes benefit from the advantage of accurate measurements on the data plane, which allows for more timely control actions on the network.

Passive measurement is one of the emerging fields for using PDPs to support operations performed at the control plane. Due to the line rate processing capability, these devices can make accurate measurements without disturbing the network behavior. Devices such as passive taps are commonly used to make an exact copy of the traffic at the monitoring points. Authors in [77] proposed a system that tracks RTTs online, using a P4 switch as a passive measurement device. Then, they apply the meter to support interception attack detection. In [78] p4-based, passive measurements of the number of flows and RTT were used to control the queue of a legacy router, improving the Flow-Completion Time (FCT) of short flows. Chen et al. used a similar approach by developing ConQuest, a tool that measures the queuing delay to identify flows that blow up the legacy router's queue.

## 2.4 Issues of large data transfers over non-dedicated networks

### 2.4.1 Coexistence among short and long flows.

Due to the distance between the sender and receiver nodes, the SBD flows often have two main characteristics: a huge data size and a high RTT. This kind of traffic is named *elephant flows*. On the contrary, general purpose traffic is short-lived flows with small volumes; hence, they are called *mice flows*. Globally, there are dedicated infrastructures optimized for SBD traffic, such as Esnet, Internet2, National Research and Education Netowrk, European Research and Education Netowork, and Janet. However, in emerging countries or small institutions, such solutions are out of budget, and therefore scientific traffic must be shared with general-purpose traffic and, therefore over non-dedicated networks. It has been proven in studies such as [79, 80, 81] that the presence of heterogeneous traffic on shared infrastructures generates a high degradation of performance in terms of utilization, latency, end-to-end throughput, and fairness index. Large flows can fill the router queues at bottlenecks, starving short flows, causing high queuing delay and packet loss and thus degradation in the quality of service [82]. On the other hand, with only a small percentage of packet loss, the performance of long flows is compromised because congestion detection and control have high convergence times due to the RTT [83].

Several solutions have been proposed to tackle the problems related to the coexistence of short and long flows. Some authors [84, 85, 86, 87] address the prioritization of packets belonging to short flows in the router queue conformation. Another strategy some authors propose is using load balancing [88] and differentiated traffic routing [89, 90, 91, 92]. The above solutions are based on the intelligent detection of elephant and mice flows. Recently, artificial intelligence has been used to identify elephant and mice flows as presented in [93, 94, 95].

### 2.4.2 Buffer Sizing

Router's buffer sizing in a communication network has been of interest to researchers due to its significant impact on the overall performance in a wide variety of network deployments. If buffers are too small, the network is prone to experience a high packet loss rate. On the other hand, if buffers are too large, the latency and complexity as well as the router's cost increase [96]. This situation has motivated studies and models for determining the appropriate amount of buffer size that should be configured, and also efforts to estimate the buffer size under a given network operation condition.

There exists a considerable body of literature around buffer sizing. Several

studies have shown that the network performance is affected by the buffer size setting in routers. In earlier works, the buffer size was configured considering the link utilization. In [97], the pervasive BDP rule-of-thumb for buffer sizing is defined. Such a rule states that the buffer size must be equal to the product between the link's capacity, $C$, and the RTT. This buffer size guarantees full utilization of the link. The results were derived by observing the behavior of a small number of simultaneous TCP flows. This rule was widely used by Internet service providers (ISPs) and manufacturers when configuring and designing routers. However, in practice, the number of flows sharing a link is usually large. Appenzeller et al. [98] incorporated this observation and proved that for $N$ desynchronized flows, the bottleneck link could be kept fully utilized with a buffer size of approximately $C \cdot RTT/\sqrt{N}$. Note that this new rule assumes that flows use long-lived TCP New Reno congestion control [99], which is based on the additive increase multiplicative decrease (AIMD) behavior. Thus, it does not apply to new congestion control algorithms based on pacing, such as BBR[100] and BBRv2[101]. Supported by the pacing technique, authors in [102] hypothesize that it is possible to obtain a remarkable network performance by further reducing the buffer size to a few dozen packets. This approach can be useful when the network administrator has full control of network resources.

Previous works showed that both the congestion control protocol and the buffer size play a crucial role in the network's performance because they are closely related. Analyses of delay-based congestion control protocols suggested that the performance of TCP-Vegas and TCP-FAST can be susceptible to the precise choice of small buffers [103]. In [104], the authors tested Orca, their own Learning-Based Congestion Control, for buffer sizes in the range of 3KB to 96MB. Results reported that Orca outperforms CUBIC [105] under any buffer size condition. Recently, Kfoury et al. [101] conducted tests for different buffer sizes, for CUBIC, BBR, BBR2, and TCP variants. They conclude that BBR and BBR2 are suitable with a small buffer size in terms of throughput and link utilization.

Experiments on buffer sizes have not been limited to controlled simulation of laboratory testbeds. In [106], Facebook's engineering team conducted a series of experiments on its backbone using real network traffic to measure the impact of buffer size reduction. The effect of buffer sizing on the Quality of Experience (QoE) metrics was measured at Netflix's network [107]. Researchers collected data using Netflix Open Connect Architecture and production traffic from users during peak hours. They observed that video QoE could degrade when both too small and too large buffer sizes are configured. The authors also state that more experiments are needed to deeply understand buffer size and CCA in video traffic performance.

When the bottleneck link parameters are not known or updated to the senders, the estimation of buffer parameters can support decision making in congestion

protocols. The Delay Friendly TCP (DFTCP) [108] performs rate control based on the estimation of the maximum size of available buffers in the routers along a network flow path. The model computes this value from average RTT and propagation delay. The parameter estimation also can be carried out from network measurements. Consequently, it is necessary to collect meaningful information from the network in a training process aiming to abstract the network's operation patterns in a general way. Authors in [109] added a bottleneck link capacity estimator to BBR to improve the performance of multipath TCP connections. Ciaccia *et al.* [110] proposed a statistical method based on traffic patterns, especially at the slow-start phase, to estimate the bottleneck's link capacity using passive measurements. The authors also train a neural network to improve the estimation accuracy of the model. ML techniques allow finding hidden patterns that favor bottleneck parameters estimation in noisy and bursty cross-traffic, multiple bottleneck links, and inaccurate time stamping. This approach is exploited in [111] where authors estimate the available bandwidth using a reinforcement learning model. The buffer size also impacts metrics such as link utilization, throughput, and resource consumption. Optimizing such metrics simultaneously was studied in [112].

### 2.4.3 Anomaly detection

One of the concerns in implementing Science DMZ is to ensure security without impacting performance. In non-dedicated networks, it is common to use firewalls, which perform detailed packet-by-packet inspection, resulting in tolerable performance degradation for general-purpose applications but not for data-intensive transmissions. Therefore, frictionless flow-based methods have been considered for implementation in Science DMZ.

There is a renew interest in using flow-based analysis to monitor and secure networks, driven by its substantial reduction in storage and CPU requirements. Hofstede et al. [113] indicate that flow-based analysis leads to data reduction of 1/2000 of the original volume, as packets are not individually processed but aggregated. Hofstede et al. [114] present flow-based detection techniques for SSH and web-application attacks. They also contrast the behavior of flows in production and lab environments. In [115], an application is developed to detect compromised hosts by using a multi-layered security detection. The first layer of detection consists of a flow-based intrusion detection, which pre-selects suspicious traffic. The second layer performs packet-based intrusion detection over the pre-filtered traffic.

Early work in entropy-based anomaly detection applied to backbone networks is presented in [116]. The proposed detection algorithms compute the entropy of IP addresses and ports. The authors showed that changes in the entropy content

are indicators of massive network events. The traffic data was collected from a large ISP backbone, namely, the Swiss national research and education network.

Nychis et al. [117] describe the advantages of entropy-based analysis of multiple traffic distributions in conjunction with each other. In particular, their work suggests that for more accurate anomaly detections, IDSs should complement the use of port and IP address distributions with other behavioral features. The different distributions are constructed by counting packets, and the entropy analysis is applied to large data sets containing thousands of active IP addresses from large networks (GEANT [118], Internet2, and others).

Berezinski et al. [119] provide a comparative study of entropy-based approaches for botnet-like malware detection. Their results indicate that, in addition to Shannon's entropy, Renyi's and Tsallis's entropies have very good detection performance. The authors also show that anomaly detection methods based on volume may perform poorly. Nowadays many anomalous network activities such as low-rate distributed denial-of-service (DDoS), stealth scanning, or botnet-like worm propagation and communication do not result in a substantial traffic volume change. Thus, they remain hidden in a traffic volume expressed by the number of packets, bytes, or flows.

Callegari et al. [120] also propose an anomaly detection system which measures the variation in the entropy associated with the network traffic. Similar to [119], different entropy definitions are used.

Homem et al. [121] propose an entropy-based technique to detect anomalies perpetrated by encapsulating IP packets carrying malware over the DNS protocol. The proposed method quantifies the information entropy of different network protocols and their DNS tunneled equivalents, and then use such quantities to discriminate normal behavior against anomalies.

The use of NAT to conveniently hide the source of malicious behavior is discussed in [122]. By using only flow information, the authors show that machine learning algorithms may identify devices behind NAT. Tracing such devices is particularly relevant when payload inspection does not provide information because encryption is used at the application layer.

## 2.5 Teaching and Research in SDN

SDN instructional initiatives have attracted increased research focus in recent years. For example, an SDN undergraduate-level course with five phases and structured around three main units (basic networking, virtualization and cloud computing, and SDN applications) was developed in [123]. Other initiatives have focused on evaluating, from an academic point of view, SDN deployment tools such as Mininet, FloodLigth, and Open Network Operating System (ONOS) [124].

Others have also focused on developing educational resources like virtual laboratories, [125], real-network platforms, [126], hands-on labs, [127], and physical testbeds, [128], for the SDN teaching-learning process. These resources provide students with a methodology to develop practical skills in production environments, albeit on a smaller scale.

Today many engineering programs have senior year Capstone Design courses, which also serve as a quality assurance requirement for accreditation processes. In such courses, students apply and integrate their acquired knowledge and skill sets from previous courses. They can also incorporate engineering standards into their implementation, along with the know-how from past work experiences. As such, many Capstone Design courses yield notable research products such as proof of concepts, system prototypes, conference papers, and journal articles [129]. They are considered a primary outcome of the learning process herein and essential for improving students' technical skills. These vehicles also provide a robust means for introducing students to the research process, which may lead to the generation of transferable knowledge for industry and society.

## 2.6 State-of-the-art Remarks

The literature review reveals much concern in deploying the Science DMZ pattern to support SBD transfers among scientific collaboration networks and to tackle the technical challenges over non-dedicated networks such as traffic control, buffer sizing and anomaly detection. Therefore, P4, in junction with data-driven approaches, emerges as a promising and flexible long-term alternative to improve the overall network performance.

# 3

# Machine Learning Controller to improve Scientific Big Data transmissions over non-dedicated Networks

This section outlines the analytical and experimental procedures and techniques used to conduct the improvement of SBD transmission over non-dedicated networks using dataplanes and machine learning control. We provide some key definitions in Section 3.1 and then the problem is stated formally in Section 3.2. Section 3.3 describes the materials and methods to provide a detailed description of the research methodology in order to make the study reproducible, and also we present the Machine Learning Controller (MLC) approach intended to improve the SBD transmissions over non-dedicated networks. The results of the tests conducted to evaluate the performance of the proposed solution under diverse scenarios are discussed in Section 3.4. The chapter concludes by presenting the contributions and conclusions of the developed solution.

## 3.1 Definitions

Before addressing the problem statement of our work, it is worth formally settling on some definitions.

**Definition 1** *A cyberinfrastructure is defined as* ***Non-dedicated Network*** *if the resources are shared among various traffic types, including long and short flows.*

**Definition 2** *An* ***Academic Cyberinfrastructure*** *is a non-dedicated network where SBD and general-purpose traffic generated by users of an academic institution coexist.*

**Definition 3** *A **Science DMZ** is a network segment of an academic cyberinfrastructure that is tailored to exchange SBD flows. The main component of the Science DMZ is the DTN which is a device responsible for transmitting scientific data to a remote destination.*

**Definition 4** *A **Campus Network** is a segment of an academic cyberinfrastructure that is largely dominated by the exchange of short-flow traffic from general-purpose applications.*

**Definition 5** *The **Network State Variables** is an abstraction of the current behavior of the network dynamics, defined as a set $y(t) = \{y_1, y_2, ..., y_K\}$ composed by $K$ measurements at a given time $t$.*

**Definition 6** *A **Control Law** $u(t)$ is a function composed by a set of network state variables $\mathcal{Y} = \{y(t), y(t-1), ..., y(t-N)\}$, a set of previous control laws $\mathcal{U} = \{u(t-1), u(t-1), ..., u(t-M)\}$, and a set of parameters $\phi$. Hence $u(t) = f(\mathcal{Y}, \mathcal{U}, \phi)$.*

## 3.2 Problem Statement

Consider the academic cyberinfrastructure simplified diagram of Figure 3.1. The campus network is divided into four main subnets. Firstly, the academic staff subnet is intended for professors and academic authorities that typically interchange emails, surf the internet, and use the internal campus platforms. Secondly, the student's subnet often uses the wireless campus network and commonly uses library resources, campus platforms, streaming, social networks, multimedia calls, and so on. Thirdly, the servers and Information Technology (IT) administration subnet supports academic databases and services and guarantees the overall network operation. The named first three segments are secured by IT administration policies implemented in devices such as firewalls and proxies that cause friction in the network traffic. Finally, the Science Demilitarized Zone (DMZ), a friction-free network segment, intended to share huge datasets with other research labs outside the network, using the non-dedicated WAN. The external resources entity summarizes all kinds of content needed to access applications of the academic cyberinfrastructure. As mentioned in Section 2, this diversity of flows provides a breeding ground for network congestion, and actions to ensure adequate service to SBD flows and short flows are required.

To deal with the above situation, the present work proposes, in summary, to assess the current state of the network and to take a relevant control action to guarantee a target rate at the DTN.

The network state variables vector $y(t)$ is composed by data-plane measurements, namely:

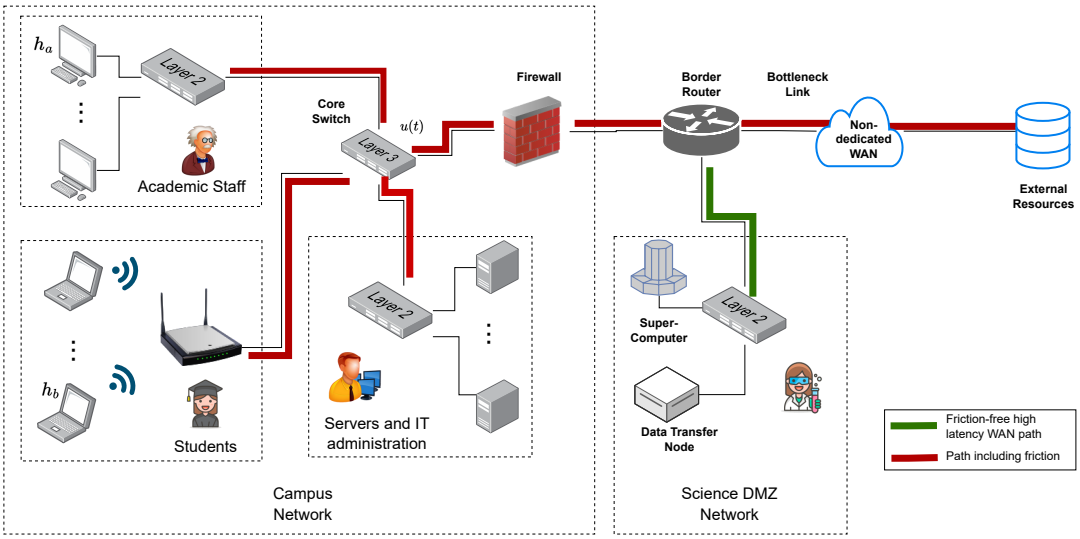



**Figure** 3.1: Academic Cyberinfrastructure.

- DTN Rate, $R_{DTN}$ [bps].
- Campus Network Rate, $R_{Campus}$ [bps].
- DTN RTT, $RTT_{DTN}$ [s].
- Campus Netowrk RTT, $RTT_{Campus}$ [s].
- Queuing Delay of DTN flows, $QD_{DTN}$, [s].
- Queuing Delay of Campus Network flows, $QD_{Campus}$ [s].
- Number of active flows, $Nf$.

Suppose the network administrator would like to set the DTN output rate at a target value $R^*_{DTN}$, as part of a desired network state $y_d(t)$. To reach the desired state, we adjust the Campus Network core switch output rate $u(t)$, following a control law described as follows:

$$\begin{aligned} g :& \mathbb{R}^{S_1} \times \mathbb{R}^{S_2} \times \mathbb{R}^{S_3} \times \mathbb{R}^q \to \mathbb{R} \\ & (y_d, y(t-n), u(t-m), \Psi) \mapsto g(y_d, y(t-i), u(t-j), \Psi) = u(t) \end{aligned} \tag{3.1}$$

with n = 1,..., N; m = 1,..., M. In the above formulation, $\Psi = \{\psi_1, \psi_2, ...\psi_q\}$ represents the set of parameters of a given control law function $g$.

Figure 3.2 shows the block diagram of the proposed solution. The control signal $u(t)$ is computed using current and past observations of the network state variables $y(t)$ and the input $u(t)$. The above leads to the following research questions: Can the network state be measured in the data plane? How to define the remaining variables of the desired state of the network? How to obtain a control law to achieve the desired state? Moreover last but not least, is it feasible to implement the proposed solution in a real non-dedicated cyberinfrastructure? The answer to these questions will be addressed in the following subsections of this chapter.

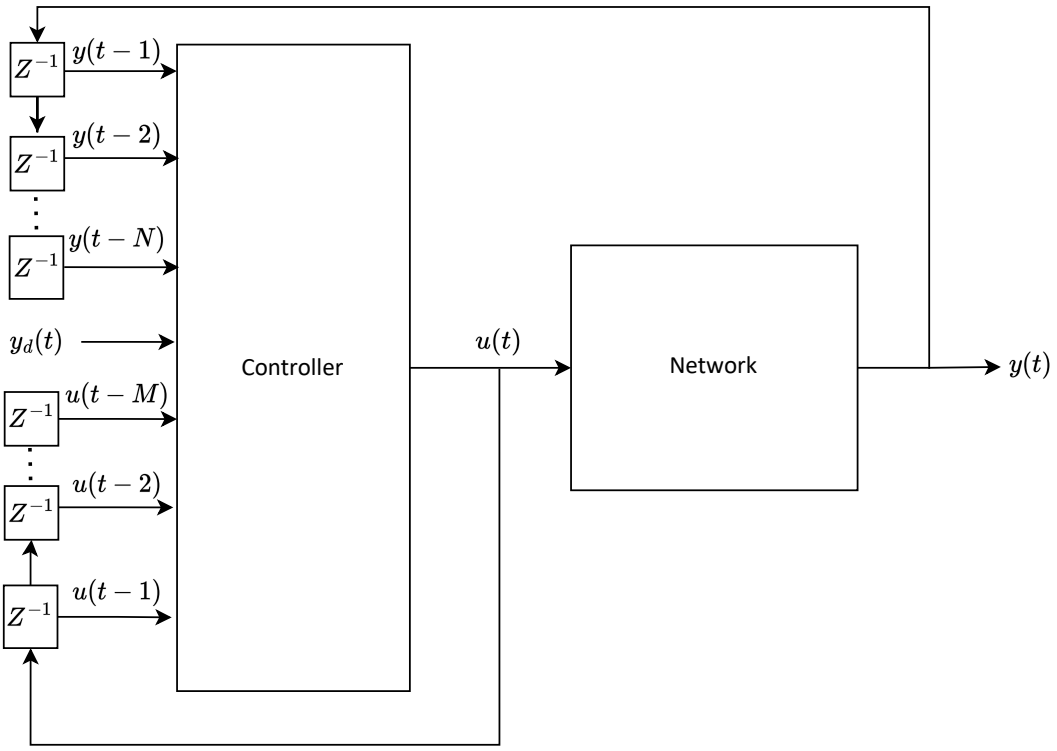



**Figure** 3.2: Rationale of Proposed Solution

## 3.3 Materials and Methods

### 3.3.1 Machine Learning Control System Architecture

According to previous experiments performed in an emulated scenario, we claim that the variation in the output rate of the Campus Network $u(t)$ affects the network state variables $y(t)$ and could be used as a control variable of the system.

Consequently, it is necessary to have a strategy to determine the control law from the system's state. Because packet loss, retransmissions, and bursts occur in a non-dedicated network, obtaining the control rule by conventional modeling methods is ineffective.

Therefore we propose to use a MLC that relies on the widely used inverse-direct controller strategy [130, 131, 132, 133]. Figure 3.3 shows the block diagram of the proposed controller.

The MLC has to fill with the following requirements:

- The ML algorithm must address a regression problem, i.e. compute a rate value at the switch from the traffic patterns obtained from the telemetry system.
- It must operate online with a computation time less than the switch output-rate update time. Because we use an Open vSwitch (OVS), for the current application, the update time is defined in 100ms.
- The algorithm must deal with the non-linearity and complexity of the network dynamics.
- It is desirable that the algorithm exploits hidden relationships between variables improving the feature extraction process.
- It must be functional from the start to guarantee the minimum flow completion time of SBD flows.

Considering the above requirements an Artificial Neural Network (ANN) can be used as controller due to:

- It can perform regression tasks using continuous output layers.
- ANN can compute online the estimated output value from the patterns in a short time depending on its architecture meeting the time requirements.
- ANN have been used in many applications and have proven to be suitable for complex and non-linear problems.
- The model can implicitly aid the feature extraction process by finding hidden patterns resulting from relationships between input variables.
- ANN could be used in pre-trained systems through supervised learning. The quality of the results depends on the amount of training data. Training the model with enough data to favor the model's generalization is possible in our solution.

The law perform for the ANN controller is formulated as follows:

$$\hat{u} = g(y_d, y(t-T), y(t-2T), u(t-T), \Psi) \tag{3.2}$$

The non-linear function $g$ and their parameters $\Psi$ are obtained by training the model with network observations every time $T$.

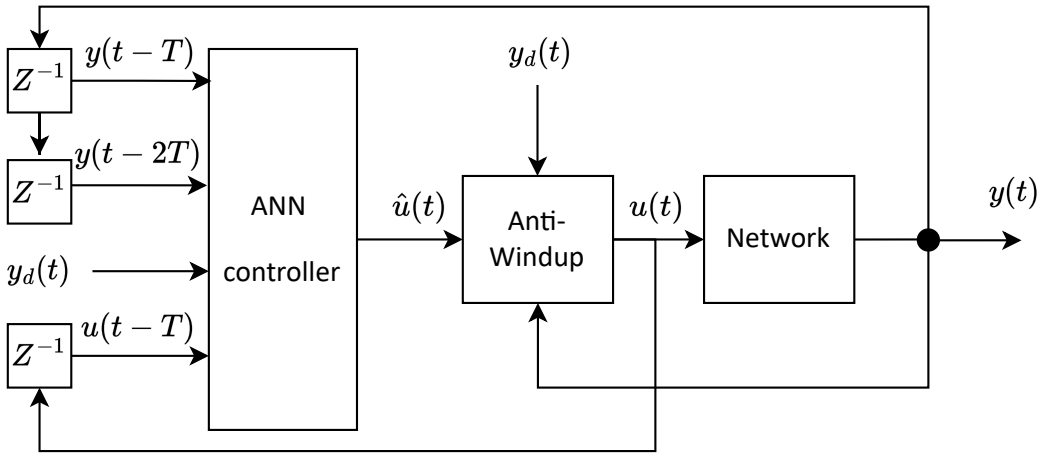



**Figure** 3.3: Machine-learning Control System Architecture.

We also included an Anti-Windup filter to reduce the error and avoid saturation in the control loop. The inverse-direct method seeks to lead the system's state to a desired state or set-point $y_d(t)$ by identifying the inverse dynamics of the system.

Figure 3.4 shows the block diagram of the Anti-windup filter, where the inputs are the error signal $e(t)$ computed by $y(t) - y_d(t)$, and the output of the ANN controller $\hat{u}(t)$. The signal $\overline{u}$ adds the contributions of the signal $\hat{u}$, the integral error, and $w(t)$. The value of $\overline{u}$ is coerced by a saturator in a $\pm\Delta u(t)$ interval. Finally, the $u(t)$ signal is set in the output interface of the Campus Network switch. The Anti-windup filter has three tuning parameters, namely: $K_p$, $K_i$, and $K_w$.

### 3.3.2 Tuning of Controller Parameters

In order to tune the controller parameters, we proposed an objective function to minimize the Integral Absolute Error (IAE) that accumulates absolute error over time. Absolute error is computed as the difference between the set-point rate $R^*_{DTN}(t)$ and observed throughput $R_{DTN}(t)$ at the DTN.

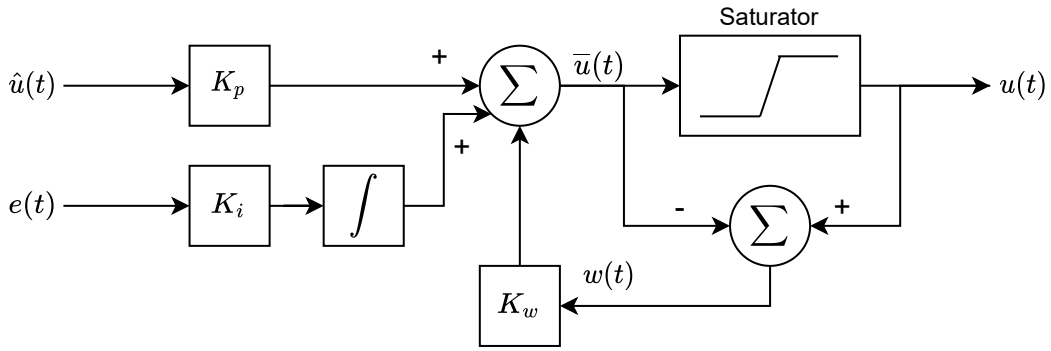



**Figure** 3.4: Anti-windup filter

$$\min_{K_p,K_i,K_w} IAE = \int_0^t \left| R^*_{DTN}(\tau) - R_{DTN}(\tau) \right| d\tau \qquad (3.3)$$
$$K_p, K_i, K_w \geq 0$$

Due to the complexity of the system dynamics, resulting from the concurrence of heterogeneous flows such as: delay, packet loss and retransmissions, it is not feasible to find a theoretical model to obtain the optimal operating values of the parameters. However, the fact that different combinations of parameters can be attempted on the testbed makes it possible to explore the solution space to determine suitable operating parameters. Since the three parameters can adopt positive continuous values, the solution space is large and therefore an effective technique is required to find the appropriate values of the parameters. Evolutionary algorithms therefore provide an effective alternative search strategy for identifying near-optimal solutions in a high-dimensional search space [134]. As shown in [135, 136, 137] Genetic Algorithm (GA) have proven to be a practical method for controller tuning, in particular PID controllers. Hence a GA was used to find the controller parameters $K_p$, $K_i$ and $K_w$.

Selecting the parameters of the GA is worthy of considering the computational limitations of the testbed. The maximum time allowed for an individual iperf3 test is 86400 s. In every computation of the fitness function or IAE inside the GA, we generate traffic from the sources to their corresponding destinations for 120 seconds using a pseudo-random $R^*_{DTN}(t)$ signal. Therefore the mating pool size and the number of generations need to be enough to offer diversity and the possibility to explore the search space and fit that maximum time restriction. The GA parameters are summarized in Table 3.1, and match with the requirements and computational limitations. For crossover and mutation probability we use

typical values. To build and implement the GA, we used the python library PyGAD [138].

Algorithm 1 shows the parameter tuning process by using the GA.

**Algorithm 1** GA for MLC parameter tuning problem.

**Require:** Generation $G$; Max Generations, $G_{max}$; Number of parents mating, $m_p$; Mating Pool Size, $m_s$; Crossover Probability, $p_c$; Mutation Probability, $p_m$

**Ensure:** Best Controller Parameters: $Kp, Ki, Kw$

$G = 0$; $P(G)$ : Initial population

Compute IAE of population ($P(G)$)

**repeat**

$M \leftarrow$ Select Parents ($P(G), m_p$)

$P_c(G) \leftarrow$ Crossover($P(G)$, $p_m$, $M$)

$P_m(G) \leftarrow$ Mutation($P_c(G)$, $p_m$)

Compute IAE of ($P_m(G)$)

$P(G+1) \leftarrow$ Selection ($P_m(G)$, steady state selection)

$G = G + 1$

Compute IAE ($P(G)$)

$K_p, K_i, K_w \leftarrow$ Get Best Solution ($P_m(G)$)

**until** $G \geq G_{max}$

**return** $K_p, K_i, K_w$

**Table** 3.1: Genetic Algorithm Parameters

| Parameter | Value |
|---|---|
| Number of generations, $G$ | 100 |
| Number of genes, $g$ | 3 |
| Number of parents mating, $m_p$ | 15 |
| Mating pool size, $m_s$ | 30 |
| Crossover probability, $p_c$ | 0.90 |
| Mutation probability, $p_m$ | 0.1 |

### 3.3.3 Experimental Testbed

Figure 3.5 shows the topology used to test our solution, and it was implemented at Cyberinfrastructure Laboratory at the University of South Carolina with real networking devices and emulated hosts. The campus network consists of 100

general-purpose sender nodes (s1, s2, ..., s100) connected by an Open vSwitch (OvS) (S1), generating commodity traffic. The Science DMZ is co-located to the campus network and consists of a DTN sender (DTN-s) and an OvS switch (S3). The campus network and Science DMZ are connected to the wide area network (WAN) through a Juniper MX204 border router (BR1). An optical passive tap is connected at each interface of the border router. Each tap replicates the observed traffic to an Edgecore 100BF-32 P4 switch, which computes network variables at line rate. The remote network consists of a Linux border router (BR2), an OvS switch (S2), 100 general-purpose receiver nodes (r1, r2, ..., r100), and a DTN receiver (DNT-r). The router BR1 is connected to the Linux Router BR2 at a link rate of 1Gbps. Therefore, this link represents the network bottleneck.

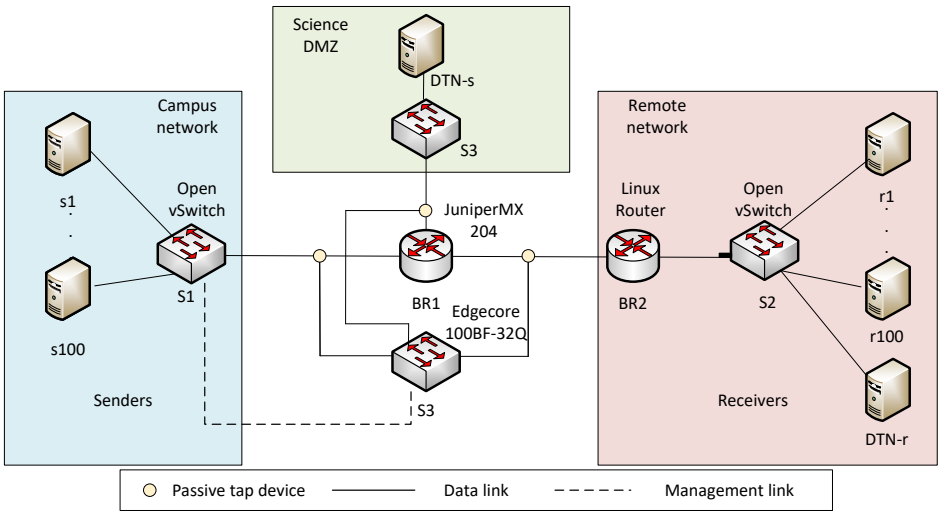



**Figure** 3.5: Testbed topology used for emulating scientific and general-purpose traffic in a non-dedicated cyberinfrastructure.

Using the Netem tool [139], we artificially introduced a 100ms delay at the link connecting the DTN-s and DTN-r nodes to emulate a high-latency Science DMZ network. Netem was also used to introduce, at the switch labeled as S1, a 10ms propagation delay for the campus network traffic. Besides, all nodes were deployed on isolated namespaces using Mininet in a Lenovo ThinkSystem SR630 server with enough computational resources.

Each sender node establishes a single TCP connection with a corresponding receiver. Instead, the DTN-s establishes multiple TCP connections to the DTN-r node, emulating the behavior of commonly used applications for scientific data transmissions like Globus GridFTP [140]. The proposed solution allows set the CCA algorithm for both the Campus Network hosts and the DTN-s.

### 3.3.4 Data Collection and Computation of Network Variables

The control system architecture shown in Figure 3.3 considers a data-driven network model. This model is defined in terms of seven network variables that jointly define the network state. To achieve this objective, it was necessary to develop codes in the data and control planes that work concurrently, as shown in Figure 3.6. We exploited the capabilities supported by the P4 switch to collect data and pre-compute such network variables online at the data plane, which are reported to the control plane. The computation of each variable is explained below.

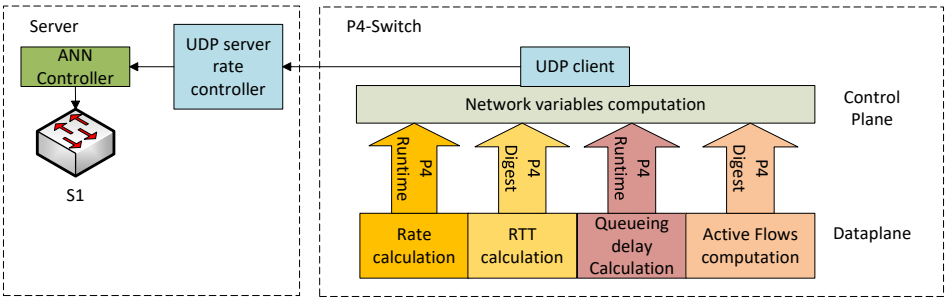



**Figure** 3.6: Functional block diagram of the solution.

#### Rate Calculation

Rate Calculation relies on codes running simultaneously at the data and control planes. At the data plane, one counter for Campus Network traffic and one for DTN's traffic is permanently updated. We use *Direct Counters*, elements available in the P4 switch architecture. As shown in Algorithm 2, when a packet enters the switch, a match-action table firstly separates the incoming traffic between the Campus Network and the DTN. Then, a counter is assigned and incremented with the packet size, in bytes, for each traffic source.

After that, as shown in Algorithm 3, the control plane polls the counters every $T$ seconds through the P4 Runtime interface and computes the difference between the current and the previous byte count per traffic, $\Delta B_{Campus}$ and $\Delta B_{DTN}$. Also, the control plane code calculates the time difference between timestamps, $\Delta t$. Thus, data rates are obtained from the ratio between each $\Delta B$ and $\Delta t$.

**Algorithm 2** Update counters at data-plane

**Require:** Packet Size $pkt.s$; Packet headers $hdr$; DTN's IP address $IP_{DTN-r}$; Remote Network IP Address Segment $IP_{RN}$; DTN counter, $C_{DTN}$; Campus Network counter, $C_{Campus}$

**Ensure:** $C_{DTN}, C_{Campus}$

```
while True do
    if hdr.ipv4.dst_addr = Longest-prefix match(DTN − r) then
        C_DTN ← C_DTN + pkt.s
    else if hdr.ipv4.dst_addr = Longest-prefix match(IP_RN) then
        C_Campus ← C_Campus + pkt.s
    end if
end while
```

**Algorithm 3** Rate calculation at control-plane

**Require:** DTN's IP address $IP_{DTN-r}$; Remote Network IP Address Segment $IP_{RN}$; DTN counter, $C_{DTN}$; Campus Network counter, $C_{Campus}$; Sample Time, $T$

**Ensure:** $R_{DTN}, R_{Campus}$

```
while True do
    Current_Bytes_Campus ← Read Counter (C_Campus)
    Current_Bytes_DTN ← Read Counter (C_DTN)
    ΔB_Campus ← Current_Bytes_Campus − Previous_Bytes_Campus
    ΔB_DTN ← Current_Bytes_DTN − Previous_Bytes_DTN
    Current_ts ← Get Timestamp(now)
    Δt ← Current_ts − Previous_ts
    R_Campus ← (ΔB_Campus × 8)Δt
    R_DTN ← (ΔB_DTN × 8)Δt
    Wait (T seconds)
end while
```

### RTT Calculation

We adapted the method proposed in [141] for the RTT computation. In general, the switch tracks TCP connections through the TCP sequence (SEQ) and acknowledgment (ACK) flags of the outgoing and incoming packets, using hash functions available in the P4 Tofino switch architecture. The RTT computation

is obtained by subtracting the timestamps of the outgoing packet and the corresponding ACK incoming packet. In our solution, we seek the outgoing and incoming flows using the passive tap at the bottleneck interface of router $R1$. Next, the RTT samples are pushed to the control plane using the P4 packet Digest interface. This is an effective method for communicating notifications from the data plane to the control plane. Digest enables the timely submission of specific fields of headers and metadata with lightweight messages that are serialized according to P4 Runtime API specification [142].

Algorithm 4 describes the operations needed at the control plane to get the RTT of DTN and Campus Network. The *flow_vector* structure tracks the active flows from the Campus Network that are transiting to the destination network and the respective estimated RTT value from the data plane. In each cycle *flow_vector* is updated using the database of flows saved in the data plane, including new flows and deleting the inactive ones using a timeout. The RTT The representative RTT value of the Campus Network, $RTT_{Campus}$ is obtained by averaging the values of the vector. Then, a smoothing function driven by an $\alpha$ parameter is applied to avoid fluctuations. The RTT of DTN flow, $RTT_{DTN}$, is retrieved directly from the digest interface.

**Algorithm 4** RTT calculation at control-plane

**Require:** Campus Network flow, $f$; DTN Flow, $f_{DTN}$; Sample Time, $T$; Vector of active flows, $flow\_vector$; Smooth Factor, $\alpha$

**Ensure:** $RTT_{DTN}, RTT_{Campus}$

```
while True do
    flow_vector ← Update from data-plane(flow_vector)
    S ← 0
    for f in flow_vector do
        S ← S + f.RTT
    end for
    Avg_RTT ← S/length(flow_vector)
    RTT_Campus ← α × RTT_Campus + (1 − α) ∗ Avg_RTT
    RTT_DTN ← Retrieve from Digest (f_DTN.RTT)
    Wait (T seconds)
end while
```

### Queuing Delay Calculation

According to [143], queuing delay is a crucial metric for early congestion detection. As shown in Algorithm 5, we compute the queuing delay as the difference between

every packet's timestamps at ingress and egress and store such value in a register. Register are stateful elements of the switch architecture that can be stored over time. We also used a match-action table to separate the DTN and the Campus Network traffic measurements.

**Algorithm 5** Update Queuing Delay registers at data-plane

**Require:** Packet, $pkt$; DTN-r's IP address $IP_{DTN-r}$; Remote Network IP Address Segment $IP_{RN}$; DNT's switch port, $DTN\_port$; Campus Network switch port, $Campus\_Network\_port$; Switch timestamp, $SW_{ts}$; Register, $Reg$

**Ensure:** $QD\_Reg_{DTN}, QD\_Reg_{Campus}$

```
while True do
    flow_id ← Hash Function (pkt.hdr)
    in_port ← pkt.intr_metadata.in_port
    if in_port = Campus_Network_port or in_port = DTN_port then
        Reg(flow_id) ← Sw_ts
    else if in_port = Bottleneck_port then
        if hdr.ipv4.dst_addr = Longest-prefix match(DTN − r) then
            QD_Reg_DTN ← SW_ts − Reg(flow_id)
            Push to Digest(QD_Reg_DTN)
        else if hdr.ipv4.dst_addr = Longest-prefix match(IP_RN) then
            QD_Reg_Campus ← SW_ts − Reg(flow_id)
        end if
    end if
end while
```

Algorithm 6 describes the tasks performed at the control plane. Every time $T$, the registers are polled from the data plane, obtaining queuing delay samples for the campus network traffic and the DTN traffic.

### Active Flows Computation

Using the approach in [144], we consider that a flow is *active* if the flow packets exceed a predefined threshold $C\_TH$ in a time less than $T\_TH$. Algorithm 7 shows the process to update the Active Flows register $AF\_Reg$. The addition or deletion of flows in that structure is reported to the control plane using Digest interfaces. The capability to read protocol-specific fields using P4 is exploited to remove flows from the registers using the FIN flag present when a TCP connection is closing. This update process is necessary not only for the accurate computation of the number of flows but also to optimize the device's memory resources.

**Algorithm 6** Queuing Delay calculation at control-plane

**Require:** Queuing Delay Register for DTN's traffic $QD\_Reg_{DTN}$; Queuing Delay Register for Campus Network traffic, $QD\_Reg_{Campus}$, Sample Time, $T$

**Ensure:** $QD_{DTN}, QD_{Campus}$

**while** True **do**

$QD_{Campus} \leftarrow$ Read Register ($QD\_Reg_{Campus}$)

$QD_{DTN} \leftarrow$ Read Register ($QD\_Reg_{DTN}$)

Wait ($T$ seconds)

**end while**

At the control plane, as shown in Algorithm 8, the digest interfaces are periodically polled at time $T$, allowing the $flow\_vector$ to be updated in the control plane. Finally, the number of active flows is calculated as the length of the flow vector.

### 3.3.5 ANN Model Training

The inverse model was obtained from an ANN model. We predict the value of $u(t)$ from the previous two network states, $y(t-T)$ and $y(t-2T)$, and the previous inverse model output $u(t-T)$. To do so, we injected a pseudo-random step signal at $u(t)$ as shown in Figure 3.7 to generate different output rates at the switch S1. We used the P4 switch as a network state sensing device, as shown on the right side of Figure 3.6. In this way, evaluating the network state for several conditions and identifying the system's inverse model is possible.

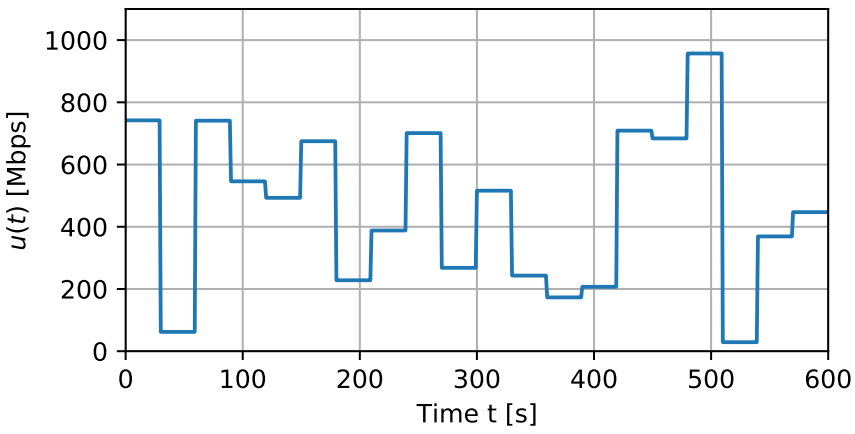



**Figure** 3.7: A realization of a signal used to train the MLC.

**Algorithm 7** Update Active Flows Register at data-plane

**Require:** packet headers, $hdr$; Switch Timestamp, $SW_{ts}$; flow identifier, $flow\_id$; Counter Register, $Counter\_Reg$; Timestamps Register, $TS\_Reg$; Active Flow Register, $AF\_Reg$; New flow digest interface, $new\_flow$; Timeout Digest interface, $timeout$.

**Ensure:** Active Flows Register $AF\_Reg$

```
while True do
    flow_id ← Hash Function (pkt.hdr)
    if flow_id in Counter_Reg then
        prev_ts ← TS_Reg[flow_id]
        TS_Reg[flow_id] ← SW_ts
        if TS_Reg[flow_id] − prev_ts < T_TH then
            if Counter_Reg[flow_id] = C_TH then
                Attach(flow_id) to AF_Reg
                Push (flow_id) to new_flow Digest
            else
                Counter_Reg[flow_id] ← Counter_Reg[flow_id] + 1
            end if
        else
            Counter_Reg[flow_id] ← 0
        end if
        if hdr.tcp.flags = FIN then
            if Counter_Reg[flow_id] = C_TH then
                Remove(flow_id) from AF_Reg
                Push (flow_id) to timeout Digest
            end if
        end if
    else
        Attach (flow_id) to Counter_Reg
        Attach (flow_id) to TS_Reg
    end if
end while
```

**Algorithm 8** Active Flows computation at control-plane

**Require:** Digest for new flows report, $new_flow$; Digest for timeout report, $timeout\_flow$; Sample Time, $T$; Vector of active flows, $flow\_vector$; flow identifier, $flow\_id$

**Ensure:** Number of active flows $AF$

```
while True do
    new_flow_id ← Retrieve from Digest(new_flow[flow_id])
    if new_flow_id then
        Attach(new_flow_id) to flow_vector
    end if
    timeout_flow_id ← Retrieve from Digest (timeout_flow[flow_id])
    if timeout_flow_id then
        Remove(timeout_flow_id) from timeout_vector
    end if
    AF = length(flow_vector)
    Wait (T seconds)
end while
```

Figure 3.8 shows the block diagram of the inverse model. The inputs or patterns are the observed network state variables $y(t)$, $y(t-T)$, $y(t-2T)$, and the past input $u(t-T)$.

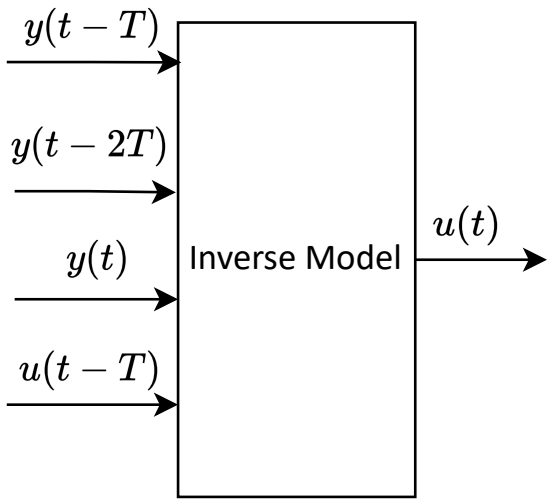



**Figure** 3.8: Inverse Model Training

Data collected was employed to train a fully connected neural network using Keras and Tensorflow ANN libraries [145, 146]. The architecture chosen is a fully connected Multi-Layer Perceptron Classifier (MLPC) neural network, as shown in Figure 3.9. MLPC is a well-established and widely used neural network architecture that offers robustness, generalization, non-linear decision boundaries, and fast convergence. Hence it matches the problem requirements. Firstly, the input layer has the same dimension as the network variables. The input variables for the neural network are the set $I = \{y(t), y(t-T), y(t-2T), u(t-T)\}$. The second layer is a hidden layer with 64 times the dimension of the inputs. The third layer is a pre-output layer with the same dimension as the input layer. Finally, the output layer has size one and represents the predicted signal u(t). In the experiments conducted, we reached a relative validation error of 1% using the cross-validation technique. Two inverse models were obtained, according to the CCA configured at the DTN, either CUBIC or BBR2. The training was done for 12,000 seconds to provide enough information to build the model.

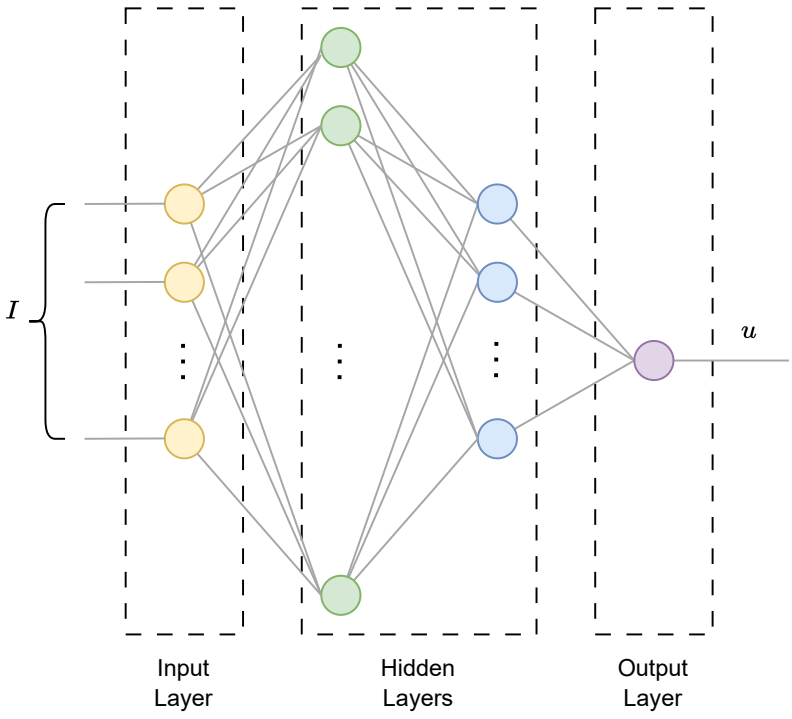



**Figure** 3.9: Neural Network Architecture used for the MLC

### 3.3.6 Control Law Implementation

The MLC control signals are carried using User Datagram Protocol (UDP) sockets on the management interface between the P4 switch and the S1 switch. The control plane generates a JavaScript Object Notation (JSON) structure containing the computed network variables $y(t)$, $y(t-T)$, $y(t-2T)$, and $u(t-T)$, every $T$ time interval. Then, it sends such a structure to the server, which in turn computes the output rate $u(t)$ feeding the trained ANN controller with the abovementioned structure. The network administrator's role is to set the desired network state $y_d(t)$ considering the target DTN-s rate and the constraints of the network.

Table 3.2 shows the values used to fill the vector $y_d$ with the desired network state. The expected value of $R_{campus}$ is the remaining capacity not required by the DTN. The RTT values are set considering the restrictions imposed in the testbed, namely $100ms$ for SBD flows and $10ms$ for general-purpose traffic. The expected queuing delays are set to 0 to minimize the bottleneck congestion. Finally, the $Nf$ is set with the last active flows computation.

**Table** 3.2: Parameters of $y_d$

| k | Variable name | Value |
|---|---|---|
| 1 | $R_{DTN}$ | Target Rate $R^*_{DTN}(t)$ |
| 2 | $R_{campus}$ | $B_{max} - R^*_{DTN}(t)$ |
| 3 | $RTT_{DTN}$ | 100 ms |
| 4 | $RTT_{Campus}$ | 10 ms |
| 5 | $QD_{DTN}$ | 0 ms |
| 6 | $QD_{Campus}$ | 0 ms |
| 7 | $Nf$ | Last $Nf$ |

### 3.3.7 Comparison Scenario

A common solution to control the campus network traffic is to use a fixed output rate. OVS switches admit Token Bucket Filter (TBF) for traffic control. Therefore, if the available bandwidth at the bottleneck is $B_{max}$, the campus output rate is set to $B_{max} - R^*_{DTN}(t)$. We named this scenario a trivial solution; it is the reference point to analyze the performance of our solution.

## 3.4 Results

In this section we provide the results of the experiments performed at the University of South Carolina's Cyberinfrastructure Laboratory with real network devices.

The tests were replicated a significant number of times in order to have statistically reliable results. Performance evaluation focuses on FCT and link utilization as metrics that quantify performance from a network perspective. However, Mean Relative Absolute Error (MRAE) is also analyzed as a metric to evaluate the algorithm's ability to track the target DTN's rate.

### 3.4.1 Controller Parameters

Using the models obtained in Section 3.3.5, we conduct the parameter tuning of the controller presented in Section 3.3.1, supported by the genetic algorithm setup show Table 3.1.

Figure 3.10 shows the convergence curve of IAE in the genetic algorithm used to tune the controller parameters when either Cubic or BBR2 are configured at the DTN. For the case of BBR2, the cumulative error difference between the actual rate and the set-point is smaller than we use Cubic. Table 3.3 summarizes the results of the parameters tuning procedure.

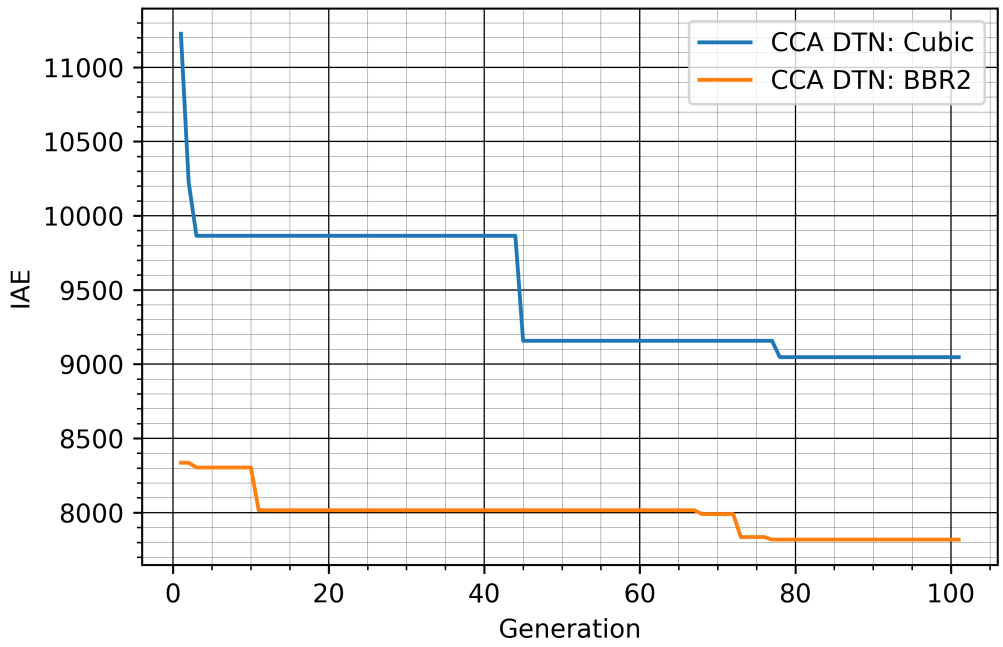



**Figure** 3.10: Convergence curve of IAE in controller parameters tuning.

**Table** 3.3: Selected controller parameters.

| DTN's CCA | Generation | $K_p$ | $K_i$ | $K_w$ |
|---|---|---|---|---|
| Cubic | 77 | 0.285 | 1.486 | 1.565 |
| BBR2 | 76 | 0.388 | 1.918 | 1.340 |

### 3.4.2 Controller performance-oriented test with multiple long flows in Campus Network

This experiment is intended to evaluate the proposed solution from the DTN's performance perspective. Therefore, the senders in the Campus Network generate long flows to their respective receivers with an induced delay of 10ms. DTN, however, generates long flow traffic with an induced delay of 100ms.

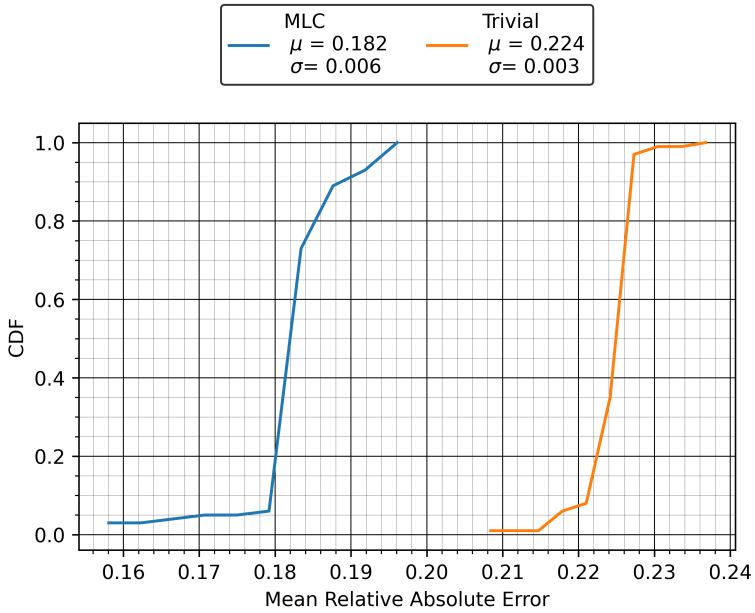
In order to test the MLC accuracy, from a control systems perspective, we compute MRAE, every $t_N$ samples as follows:

$$MRAE = \frac{1}{t_N} \sum_{t=1}^{t_N} \left| \frac{R^*_{DTN}(t) - R_{DTN}(t)}{R^*_{DTN}(t)} \right| \tag{3.4}$$

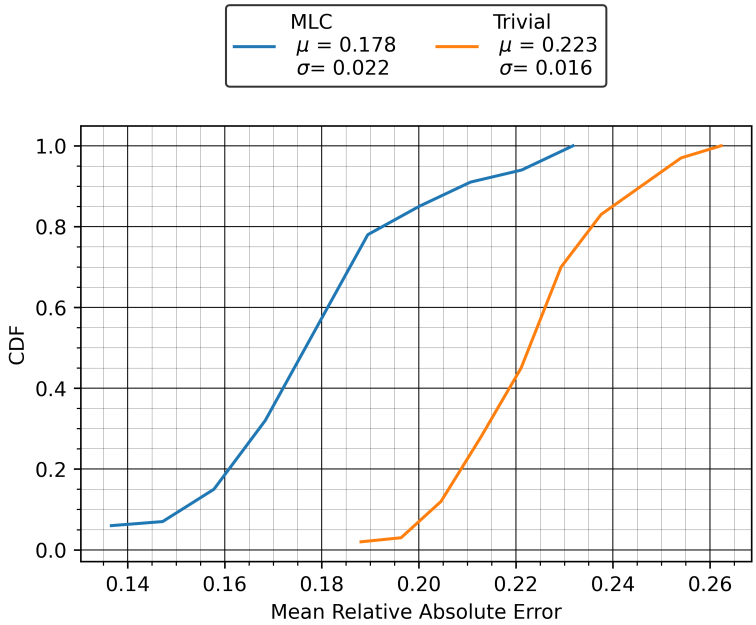
In this experiment, $t_N = 120$, and due to $\Delta t = 1s$ each trial takes 120s. We carried out 100 trials for each MLC. Cumulative Distribution Functions (CDFs) of MRAE are shown in Figure 3.11. The implementation of MLC, in general, outperforms the trivial solution in terms of MRAE, with an average reduction of 4.1% and 4.5% for the MLCs of Cubic and BBR2 respectively.

### 3.4.3 DTN's performance oriented test with multiple long flows in Campus Network

While the proposed controllers adequately track the reference, evaluating the solution's performance in terms of network metrics is essential. FCT is the time elapsed between sending the first packet and receiving the last packet for a given TCP connection. This experiment evaluates the FCT for a 10GB data transmission from DTN-s to DTN-r. The experiment was repeated 100 times for each MLC, obtaining the CDFs shown in Fig.3.12. An average reduction, against the trivial solution, of 12.932 s and 39.042s of FCT for Cubic and BBR2 controllers were found, respectively. The previous means a FCT reduction of 7.4% in the case of Cubic's MLC, and an average reduction of 21.71 % using the BBR2's MLC. We also noted that 100% of the attempts with the proposed controller obtained a lower FCT than the trivial solution for the BBR2 case. The results imply that transmitting more scientific data in a given time interval with the pro-




(a) DTN: Cubic




(b) DTN: BBR2

**Figure** 3.11: Mean Relative Absolute Error

posed MLCs is possible. We also compute the bottleneck Link Utilization $\rho$ using

the data plane measurements of the previous experiment. This value is obtained at time $t$, through the sum of the flow rates, including traffic generated by the DTN $R_{DTN}$ and the campus network hosts $R_h$, divided by the maximum link bandwidth $B_{max}$, as follows:

$$\rho(t) = \frac{R_{DTN}(t) + \sum_{h=1}^{100} R^h(t)}{B_{max}} \tag{3.5}$$

Fig. 3.13 presents the CDFs of bottleneck link utilization for the MLCs and their respective trivial comparison scenario. We found similar behavior in the two situations, with an average negligible decrease of 0.57% and 0.65% in the $\rho$ of the proposed solutions.

### 3.4.4 Performance of short flows sharing the bottleneck with long flows

The present experiment evaluates the FCT of short flows when sharing the bottleneck link with long flows for the MLCs proposed. The Weibull heavy-tailed distribution has been widely used to model the behavior of short flows on the Internet [147]. The probability distribution function of a Weibull random variable is defined as follows:

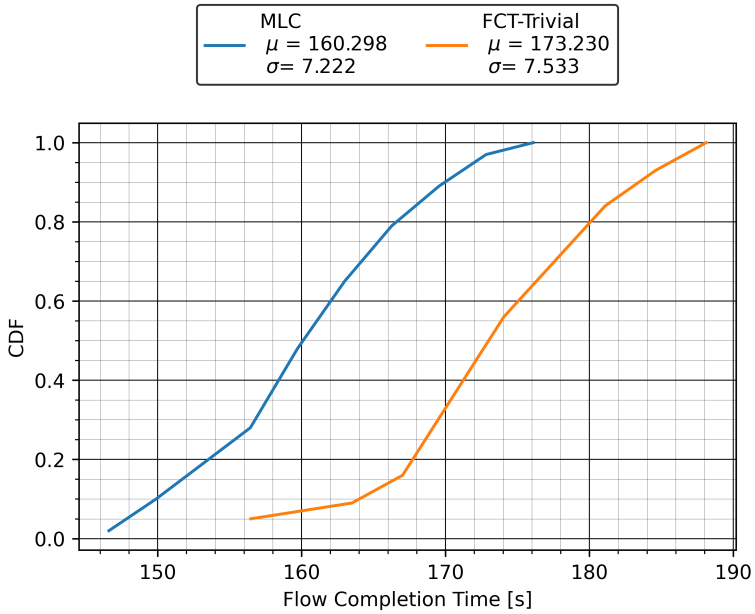
$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-\left(\frac{x}{\lambda}\right)^k} & \text{if} \quad x \geq 0 \\ \\ 0 & \text{if} \quad x < 0. \end{cases} \tag{3.6}$$

Where $k$ and $\lambda$ are called the *shape* and *scale* parameters, respectively. To model the short flows, we use two variables. Firstly, the Inter-departure Time (IDT) measures the difference between the departure time of one packet and the next. Secondly, the Flow Size (FS) is the number of bytes occupied by each flow. We use the SourcesOnOff tool [148] that allows us to generate flows with given distributions for IDT and FS. Table 3.4 summarizes the parameters that describe the short flows from Weibull distributions in the experiments.

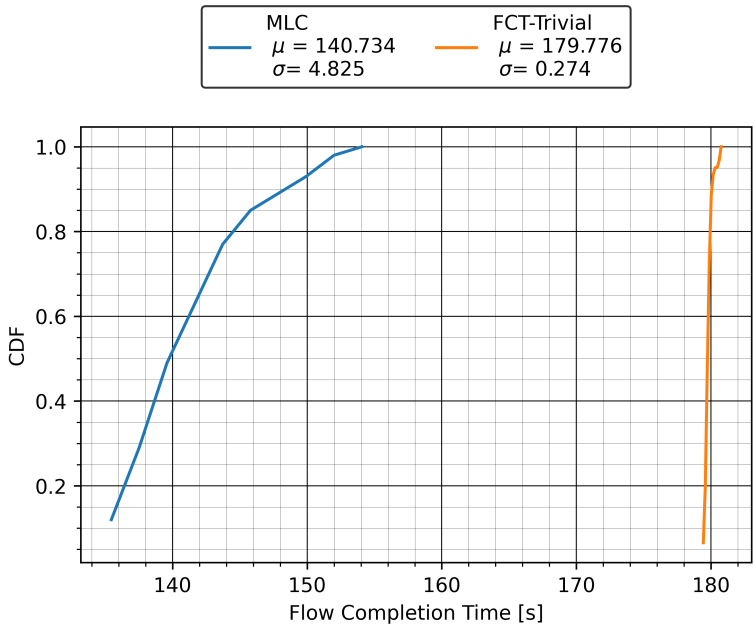
**Table** 3.4: Selected controller parameters.

| Variable | $k$ | $\lambda$ | min | max |
|---|---|---|---|---|
| IDT | 0.5 | 20ms | 1ms | 100ms |
| FS | 0.5 | 100kB | 10kB | 10MB |

The setup is similar to the one presented in Section 3.4.3, with the difference that one of the senders generates short flows. By capturing the packets during



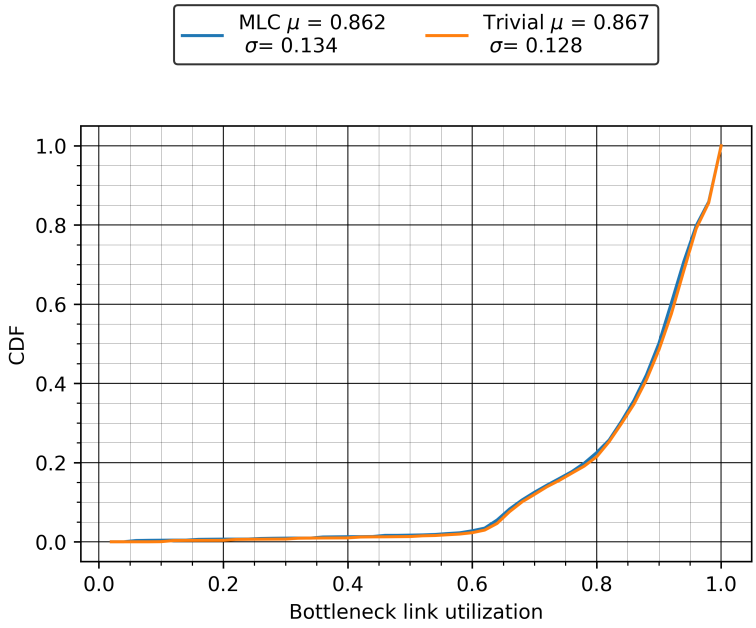

(a) DTN: Cubic



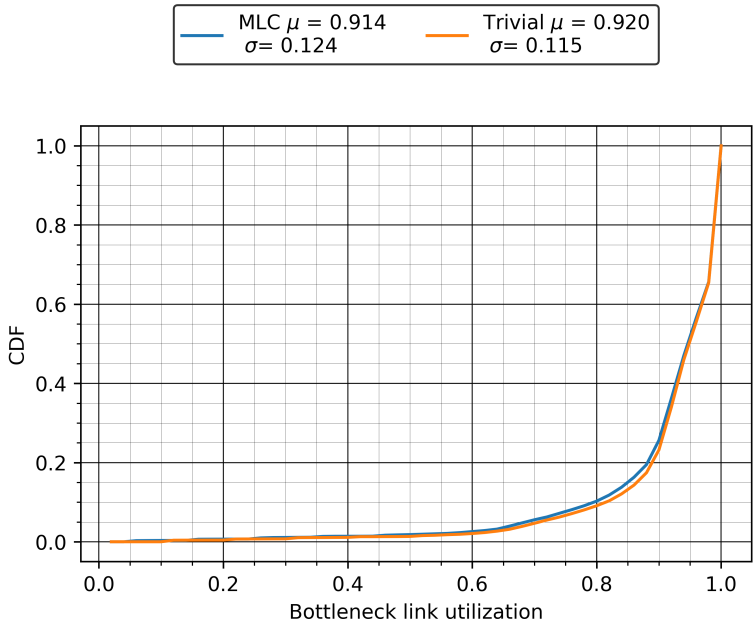

(b) DTN: BBR2

**Figure** 3.12: DTN's Flow Completion Time

600s and processing them with the *tcptrace* tool, we obtained the CDFs of figure




(a) DTN: Cubic



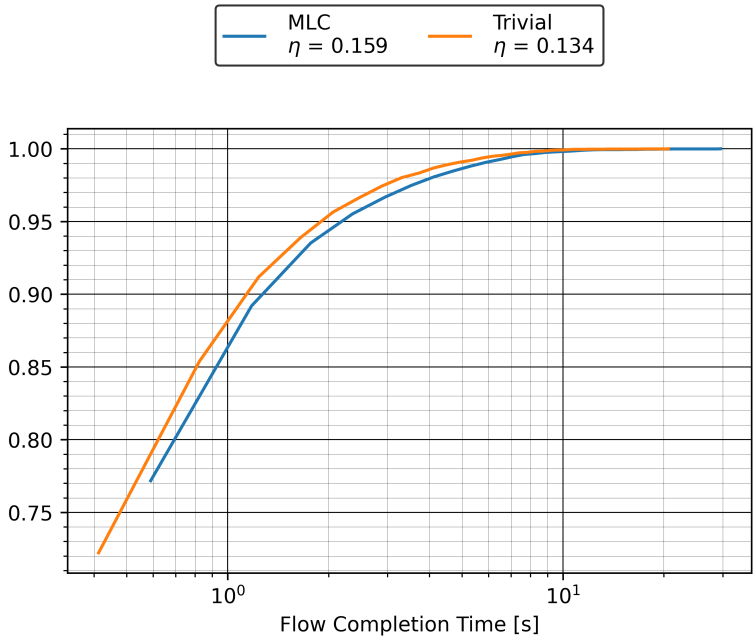

(b) DTN: BBR2

**Figure** 3.13: Bottleneck link utilization

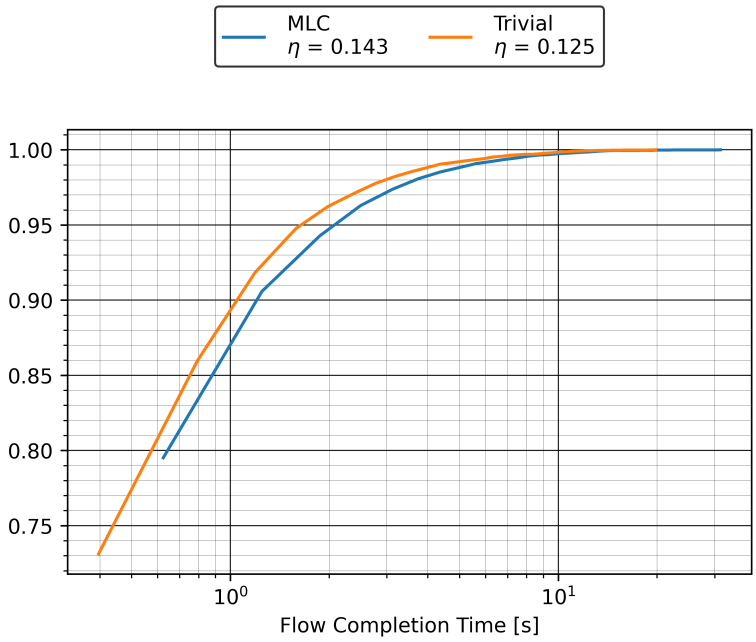
3.14. When we compare the CDFs of the proposed MLCs concerning the trivial solution, we devise that the curves are too close. There is an increment of 18.7% and 14.4% in terms of the medians $\eta$ of $\rho$ for the Cubic and BBR2 controllers, respectively. Considering the absolute variations, $\eta$ for the MLCs trained with Cubic and BBR2 have 25ms and 18ms increments, which are negligible for most campus network applications.

## 3.5 Conclusion

We demonstrated the potential of data plane devices and machine learning algorithms for improving rate control in non-dedicated networks. By posing a novel MLC approach to adjust traffic flows dynamically, we could enhance the FCT of data-intensive scientific flows coexisting with short flows generated in the campus network. In particular, our study highlights the importance of adaptive rate control, which can respond online to changing network conditions and traffic demands.




(a) DTN: Cubic




(b) DTN: BBR2

**Figure** 3.14: Flow Completion Time for short flows.

# 4

# Bottleneck Router's Buffer Size estimation for heterogeneous TCP sources

In this section we explore the problem of buffer size estimation in heterogeneous networks. The proposed approach relies on the use of end-to-end measurements to infer the operating regime of the bottleneck router in a non-dedicated one with heterogeneous hosts. Section 4.1 presents the problem formulation and the proposed solution through a classifier. Section 4.2 presents the methodology used in the study based on the framework of a supervised learning problem. The discussion of the results is addressed in Section 4.3, while Section 4.4 mentions the main contributions of the proposed approach.

## 4.1 Problem Statement

Consider the topology in the Figure 4.1, consisting of a source network and a destination network that are connected via a WAN. The source network is composed of the set $S = \{s_1, s_2, ..s_n\}$, $i = 1, ..., n$ while the destination network is composed of the set $R = \{r_1, r_2, ...r_n\}$, $i = 1, ..., n$. Each sender $s_i$ establishes a connection with a $r_i$, The link between the networks has a bottleneck that represents the portion of the network where all the flows converge and thus where the congestion is generated.

The bottleneck is characterized by the Bandwidth Delay Product (BDP), which corresponds to $BDP = BW \times D$, where $BW$ represents the maximum channel capacity and $D$ the nominal delay at the bottleneck. In figure 4.1 BR1 represents the routing device associated with the bottleneck and has a buffer size $BS = k \times BDP$, where $k \in \mathbb{R}^+$. If we constrain the $k$ values in a finite set $K = \{k_1, k_2, ..., k_j\}$, then there are $j$ buffer size regimes.

Now consider the set $C = \{c_1, c_2, ...c_m\}$, $m = 1, .., n$ which represents the $m$

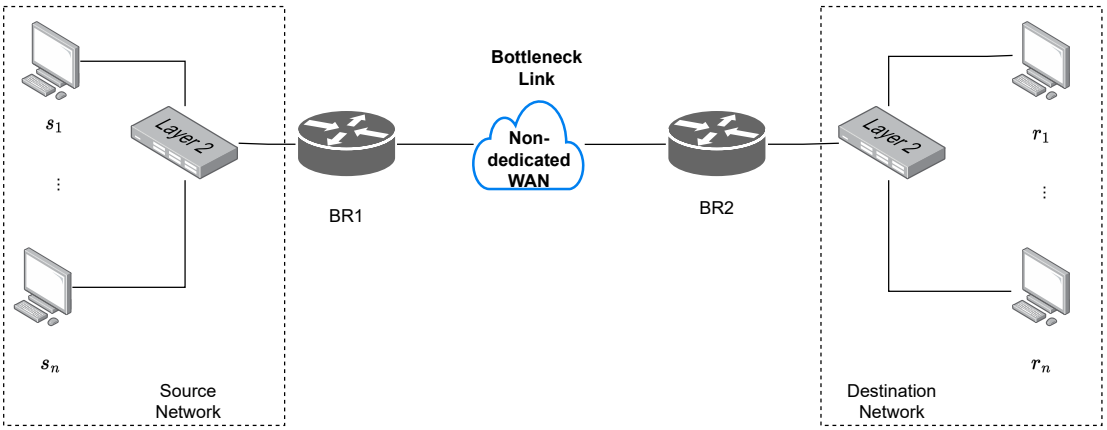



**Figure** 4.1: Non-dedicated network with heterogeneous sources

possible types of senders according to a Congestion Control Algorithm (CCA) algorithm. If $m > 1$ we claim the sources are heterogeneous. Finally, we denote $P = \{p_1, p_2, ...p_l\}$, $p = 1, ..., l$, the set of patterns that describes the network state thought implicit measurements.

Considering the above, the following research question arises: How to estimate coarsely the bottleneck router's buffer size using a model that relies on a set P of indirect measurements in a network with n sender-reciever pairs and m classes of senders? The classification problem is stated as follows:

$$\begin{aligned} f :& \mathbb{R}^l \times \mathbb{R}^q \to \mathbb{R} \\ & (P, \Phi) \mapsto f(P, \Phi) =: k_i, \quad \text{with} \quad P = P(s_i, c_i, c_m), \end{aligned} \tag{4.1}$$

In the above formulation, $s_i \in S$, $r_i \in R$, $c_m \in C$ and $\Phi = \{\phi_1, \phi_2, ...\phi_q\}$ represents the set of parameters of a given classifier function $f$.

## 4.2 Materials and Methods

To carry out the buffer size estimation, we used the workflow based on the supervised Machine Learning (ML) approach depicted in Fig. 4.2. In brief, the methodology is divided into four stages, namely: data collection, feature engineering, training, and evaluation.

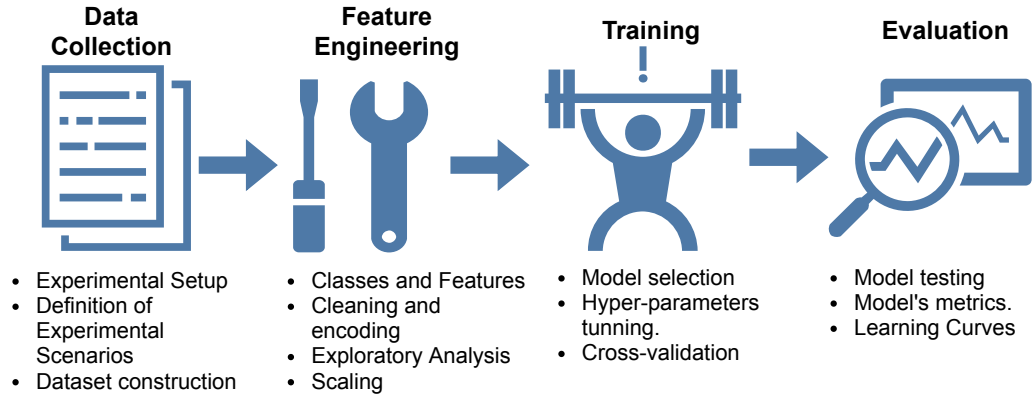



**Figure** 4.2: A workflow for the coarse buffer size coarse estimation

### 4.2.1 Data Collection

#### Experimental Setup

Figure 4.3 shows the topology used to collect data. The testbed is an extension of the well-known dumbbell topology. The working scenario consists of 100 sender nodes, labeled as h1 to h100, and 100 receiver nodes, labeled as h101 to h200. For simplicity, sender and receiver nodes were emulated on isolated namespaces using Mininet and deployed in two separate servers to ensure sufficient computational resources. During the experiments, each sender node establishes a single Transmission Control Protocol (TCP) connection with one receiver node. TCP Cubic, Bottleneck Bandwidth and Round-trip (BBR), and Bottleneck Bandwidth and Round-trip version 2 (BBR2) congestion control algorithms were configured at the sender nodes to introduce heterogeneity in the way senders react and consequently in the measurements. The Netem tool was used at the switch labeled as S1 to configure different propagation delays and random packet loss rates. The propagation delay parameter was set to 20 ms, while packet loss percentages were set to 0%, 1%, 2%, and 3%. We established the bottleneck link rate in 1Gbps at the interface linking routers labeled as R1 and R2. Lastly, we qualitatively describe the bottleneck link-state coarsely using a finite set of values for the buffer size, expressed in BDP units. Considering the above setup, BDP is the product of 1Gbps and 20ms (i.e., $2.5 \times 10^6$ bytes). Thus, the buffer size at R1 was configured at 0.01 BDP, 0.1BDP, 1BDP, 10BDP, and 100BDP representing a very small, a small, a moderate, a large, and a very large buffer sizing regime. Hence, K = {0.01, 0.1, 1, 10, 100}. In all regimes, the buffer size is greater than the expected size of a TCP segment.

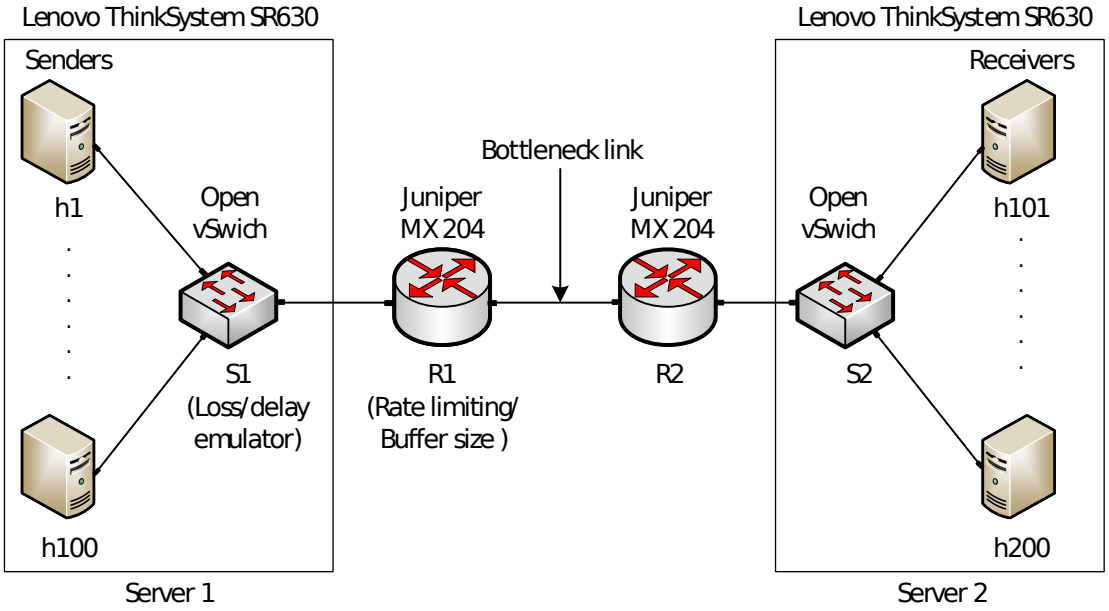



**Figure** 4.3: Topology used for data collection.

### Definition of experimental scenarios

Here, an experimental scenario represents a network condition, given by a packet loss level and buffer a size regime, evaluated by exchanging information between the sender and receiver nodes. Table 4.1 lists the settings for the experimental scenarios evaluated in the study. We used a nearly even distribution of congestion control algorithms assigned to hosts and a no packet loss scenario for ML training and validation purposes. We employed an unbalanced distribution of such congestion control algorithms to test our coarse estimation method for the bottleneck router buffer size. We also used four different packet loss levels during testing to assess whether our inference method can generalize its predictions under unseen training scenarios. In total, five scenarios were defined for training and validation, and 20 scenarios for testing. For each scenario, the set of patterns P is composed by, Round Trip Time (RTT) and Congestion Window (CWND). We got the above measurements using iPerf3. Each scenario was executed for 120 seconds.

### Construction of the ML Dataset

Measurement traces were processed to compute, at each host and for every scenario, the following time-averaged metrics: throughput, RTT, and CWND at the sender side. Also, we labeled the data with its corresponding CCA and buffer size regime. The training and validation dataset was built by joining the data provided by the scenarios with no packet loss and the five defined buffer size regimes in

**Table** 4.1: Experimental Scenarios

| **Phase** | **Host's CC Algorithm** | **Packet Loss (%)** | **Buffer Size (BDP)** |
|---|---|---|---|
| Training and validation | 34 CUBIC<br>33 BBR<br>33 BBR2 | 0 | 0.01, 0.1, 1, 10, 100 |
| Test | 60 CUBIC<br>25 BBR<br>15 BBR2 | 0, 1, 2, 3 | 0.01, 0.1, 1, 10, 100 |

a balanced distribution of host CCAs. The remaining scenarios with unbalanced CCA distribution were used to build four test datasets, one for each packet loss level.

### 4.2.2 Feature Engineering for the Coarse Buffer Sizing Regimes

#### On the Classes and Features

Using an ML approach, we define a representation space for the coarse buffer sizing regime in terms of explanatory variables termed as features. From the networking literature, relevant feature variables for describing end-to-end connections are the average throughput (in Mbps), the RTT (in ms), the CWND (in MB), and the congestion protocol. The average throughput, the average RTT, and the CWND features are real-valued, while the congestion protocol feature is categorical. Such end-to-end features are used, in turn, as proxies for the latent space representing the coarse buffer sizing regimes at the bottleneck router. Thus, we can state the problem of inferring the buffer sizing regime as a multi-class discrimination problem where each class is mapped onto a single coarse buffer size configured at the router.

#### Dataset Cleaning and Enconding

Firstly rows with missing values due to data corruption or record data failures were deleted from the dataset to yield robust classification models. In addition, the categorical values associated with the congestion control protocols and buffer size regimes were encoded using integer values to match the requirements of the classification methods used in the present study.

### Exploratory Analysis of Feature Variables

Table 4.2 lists the descriptive statistics for the average throughput, the RTT, and the CWND feature variables in the training ad validation dataset. Features exhibit a large variance, as evidenced by both the extreme values and the coefficient of variation. We also estimated the empirical probability distributions for each feature by computing their histograms. In such calculations, we used the Freedman-Diaconis rule for binning. Histograms in Fig. 4.4 show that classes (buffer sizing regimes) overlap at several intervals in the feature space. Therefore, it can be noted that variables cannot be used independently to infer the coarse buffer size at the bottleneck router. Figure 4.5 shows the correlation matrix between the features variables. It can be noted that features do not exhibit a high degree of correlation, meaning that no feature variable should be discarded during training.

**Table** 4.2: Descriptive Statistics of the Training and Validation Datasets for the non-categorical feature variables.

| **Stat** | **Throughput (Mbps)** | **RTT (ms)** | **CWND (MB)** |
|---|---|---|---|
| Mean | 10.419 | 138.954 | 0.906 |
| Standard Deviation | 11.220 | 162.415 | 2.341 |
| Min value | 0.800 | 21.802 | 0.007 |
| Max value | 110.976 | 457.592 | 37.732 |
| Coefficient of variation | 1.077 | 1.17 | 2.583 |

### Scaling

Since data ranges for the feature variables are very different, we normalized the non-categorical variables in the range [0,1] to overcome the units' differences and contribute to the model generalization.

## 4.2.3 Model Training

Six classic and state-of-the-art classifiers were trained to infer the coarse buffer sizing regimes. The trained classifiers are the Least Squares (LS), the Gaussian Naive Bayes (GNB), the Linear-kernel Support Vector Machine (SVM), the Radial Basis Function (RBF) kernel SVM, the K-Nearest Neighbors (KNN), and the Multi-Layer Perceptron Classifier (MLPC). Each algorithm provides insights about the data used in the problem addressed. GNB helps determine whether features are

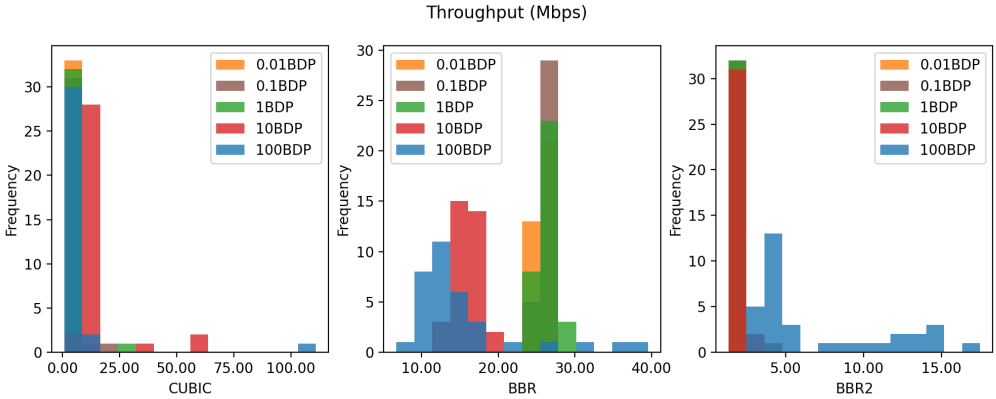
independent. LS and Linear-kernel SVM are used to test if the classes are linearly separable. RBF SVM captures complex and non-linear relationships with a small amount of data. KNN and MLPC are suitable where the data is not linearly separable, and the decision boundary is complex. Here, we use the LS classifier as a baseline instead of using the typical Zero Rule or random picking classifier.

We used a stratified shuffle split approach for cross-validation with five iterations and a test-size of 20%. To tune the hyperparameters associated with each classifier, we carried out an exhaustive search over a grid of candidate parameters. Classification Accuracy (Acc) metric was chosen as the primary performance measure. Acc is defined as the ratio of correct predictions to the total number of input dataset instances. A correct prediction means that the algorithm is able to determine the state of the network in terms of bottleneck router's buffer size from end-to-end measurements. Table 4.3 lists both the parameters and values used to train the classifiers. Training and validation were carried out in Python, exploiting the benefits of the Scikit-learn library [149]. In summary, hyperparameter tuning was employed to achieve better classification results, while cross-validation was used to avoid overfitting and favor the models' generalization.

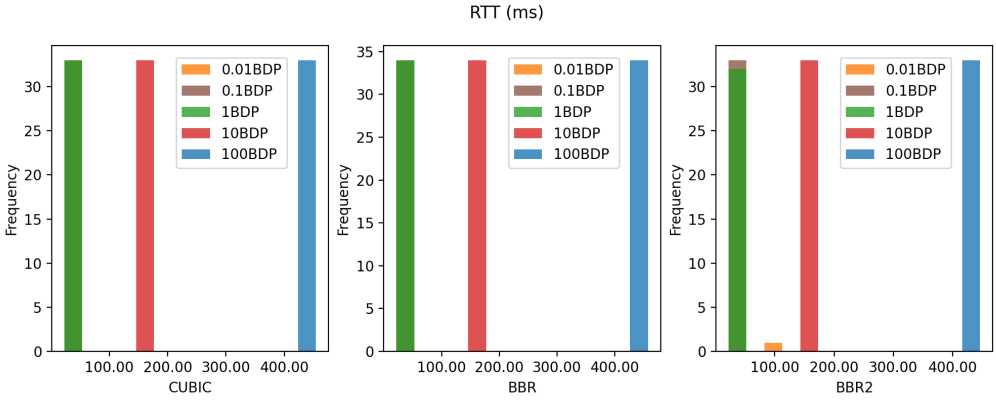
**Table** 4.3: Training values for the hyperparameters at each classifier.

| **Classifier** | **Parameters** | **Range** | **Spacing** | **Values** |
|---|---|---|---|---|
| SVM-Lineal | C | $[1, 10^5]$ | Log | 6 |
| SVM-RBF | C | $[10^{-3}, 10^{10}]$ | Log | 20 |
| | gamma | $[10^{-9}, 10^3]$ | Log | 20 |
| KNN | n_neighbords | [2,9] | Linear | 8 |
| MLPC | max_iter | [1000, 4000] | Linear | 4 |
| | alpha | $[10^{-1}, 10^{-4}]$ | Linear | 4 |
| | hidden_layers | [1,14] | Linear | 14 |
| | random_state | [1,9] | Linear | 9 |

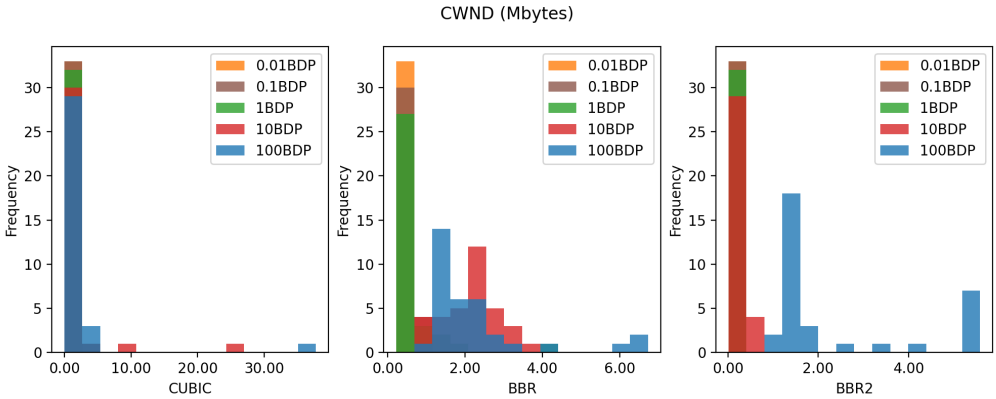
Next, we present the results that support our claim that we can infer, at the sender side, the coarse buffer size of a bottleneck link, using the summarized information collected from network measurements and processed with state-of-the-art ML algorithms.




(a) Average end-to-end throughput




(b) Average RTT




(c) (c) Average CWND

**Figure** 4.4: Histogram of the non-categorical feature variables

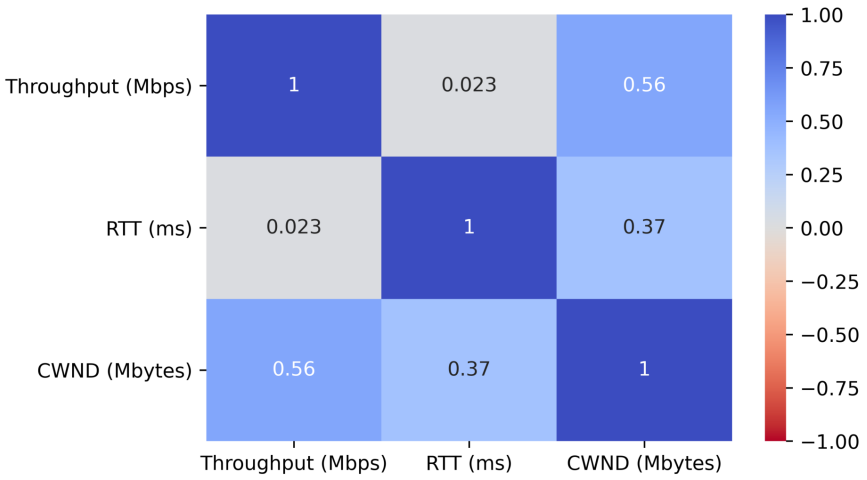



**Figure** 4.5: Correlation matrix for the non-categorical feature variables.

## 4.3 Results and Evaluation

Table 4.4 summarizes the performance of classifiers designed in this paper at the model validation stage in terms of Acc. We provide the best values for the trained models' hyperparameters, that is, those with which the maximum ACC was obtained.

**Table** 4.4: Summary of the classification results for all the coarse buffer sizing estimators.

| **Classifier** | **Validation** | | | **Test** | | | |
|---|---|---|---|---|---|---|---|
| | **Acc** | **Parameter** | **Best** | **Acc No Loss** | **Acc Loss 1%** | **Acc Loss 2%** | **Acc Loss 3%** |
| LS | 0.45 | | | 0.45 | 0.40 | 0.40 | 0.40 |
| Gaussian NB | 0.8 | | | 0.98 | 0.80 | 0.77 | 0.75 |
| SVM-Linear | 0.86 | C | 10 | 0.89 | 0.99 | 0.97 | 0.89 |
| SVM-RBF | 0.99 | C | $10^{10}$ | 0.99 | 1.0 | 0.97 | 0.92 |
| | | gamma | $2.63 \times 10^{-5}$ | | | | |
| KNN | 0.98 | n_neighbords | 2 | 0.98 | 0.98 | 0.96 | 0.91 |
| MLPC | 0.98 | max_iter | 3000 | 0.93 | 0.95 | 0.93 | 0.90 |
| | | alpha | 0.1 | | | | |
| | | hidden_layers | 13 | | | | |
| | | random_state | 4 | | | | |

We comment first that our problem's baseline classifier performance is $Acc = 0.45$. (We note that for a five-class Zero Rule detector, its performance is 0.2.). Out of the five trained classifiers presented here, the lowest performance at the validation stage was obtained by LS classifier. Such classifier achieved a classification accuracy of 0.45. This is attributed to its inability to separate the classes in the feature space domain. The highest performance was obtained by the RBF-based

SVM classifier, whose classification accuracy of very close to 1 during validations.

Figure 4.6 compares the learning curves of proposed classifiers during the training and validation stages as a function of the number of examples provided to the model. In all classifiers, we obtain a minimal gap (i.e. less than 0.05 of Acc) between training and test curve. This finding indicates that models deal correctly with underfitting and overfitting issues and also that the training dataset was representative to provide sufficient information for model construction. It is observed that as data are added to the linear SVM and RBF training set, they keep a remarkable and constant performance, and these need a reduced number of instances to reach a suitable performance. We also note that GNB-based and SVM lineal classifiers reduce its performance as the number of training samples increases. This behavior is attributed to the inaccurate assumption about the feature variables being jointly independent, as stated depicted in Fig. 4.5.

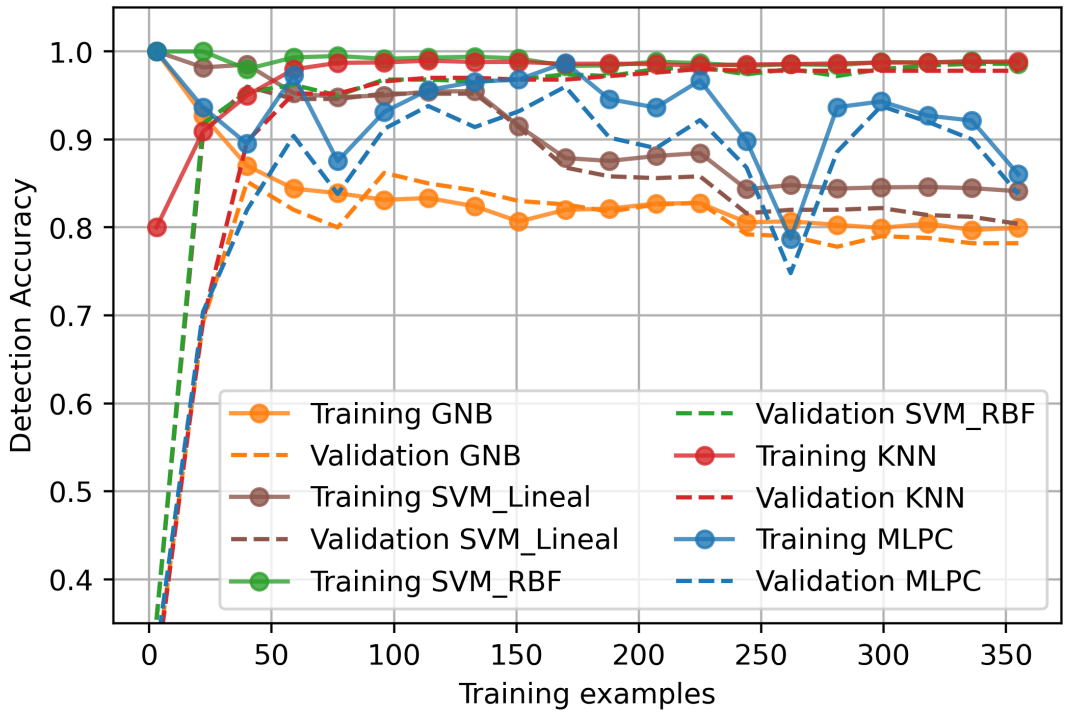



**Figure** 4.6: Learning curves of the classifiers.

To assess our inference models' ability to generalize their results, we inferred the coarse buffer sizes using test data not used during the training and validation stages. In the first test scenario, which considers imbalanced CCAs and no packet loss conditions, the classifiers keep similar performances to those presented during the validation stage. These results show that the feature variables used here are suitable to describe a particular buffer size condition. In the remaining three

test scenarios, which also consider imbalanced CCAs yet introduce different levels of packet losses, we observe again that the classifiers' performance decreases in small fractions. We highlight that the linear-kernel SVM and the MLPC model generalize their results fairly well, since their classification accuracy decreases less than 3% concerning the no pack loss test scenario. Above this level of packet loss, the accuracy of the models degrades significantly. Although a packet loss level greater than 3% is uncommon in our network scenario, this is a situation that can occur, for example, if wireless links are present in the access network. Therefore future works can be conducted in that direction.

## 4.4 Conclusion

We provide a suitable method to coarsely estimate the buffer size regime at the bottleneck link, following the supervised ML approach. Our findings indicate that average values of throughput, RTT, and CWND and the CCA used by the sender-side are useful features for building buffer size inference models. Out of our six trained models, the RBF-based SVM classifier outperforms the remaining ones in estimating the coarse buffer size. This finding is attributed to its ability to adapt to the classes' non-linear separability evidenced in the exploratory analysis. Furthermore, this classifier exhibited a robust performance in the presence of heterogeneous unbalanced sources when packet losses are induced in the network. Regarding the transfer of Scientific Big Data (SBD) flows over non-dedicated networks, knowing in advance the buffer size at the bottleneck can help to adjust the Data Transfer Node (DTN) configuration including parameters of the congestion control algorithm.

# 5

# Flow based characterization of an Academic Cyberinfrastructure for Intrusion Detection

Science DMZ networks can be vulnerable to various security issues. Some of the highest security issues associated with Science DMZ networks include:

- *Limited access controls*: Science DMZ networks are typically designed to provide high-speed data transfer, which means they may have limited access controls. This can make it easier for attackers to gain unauthorized access to the network and its data.
- *Friction*: Many Science DMZ networks do not use encryption for data in transit or firewall devices, making it easier for attackers to intercept and modify network traffic.
- *Inadequate monitoring*: Science DMZ networks can generate large amounts of network traffic, making it difficult to detect and respond to security incidents promptly. Inspecting every packet is unfeasible due to affectation on performance.
- *Update management*: Science DMZ uses no commonly specific-purpose hardware and software, making it challenging to keep up with patching and updates. The above can leave vulnerabilities open to exploitation by attackers.

Regarding the above, novel security solutions need to be developed to support high-performance SBD transfers. In this chapter, an anomaly detection system is proposed. Instead of inspecting individual packet, aggregate statistics of traffic flows are characterized to obtain normal conditions and hence suggests operational anomaly ranges based on the entropy measure. The Section 5.1 presents the target context and the problem description to be addressed. Section 5.2 presents the

materials and methods needed to develop the solution. Section 5.3 presents the most relevant findings of the flow characterization in the campus network. Finally, the conclusions of the study are presented in section 5.4.

## 5.1 Problem Statement

Consider the topology shown in Figure 5.1. The academic network behind the Border Router establishes connections with hosts in a non-dedicated network such as the Internet. Because of the academic network's private addressing and the scarcity of Internet Protocol (IP) addresses on the non-dedicated network, hosts on the academic network use the Network Address Translation (NAT) service to access resources on the non-dedicated network.

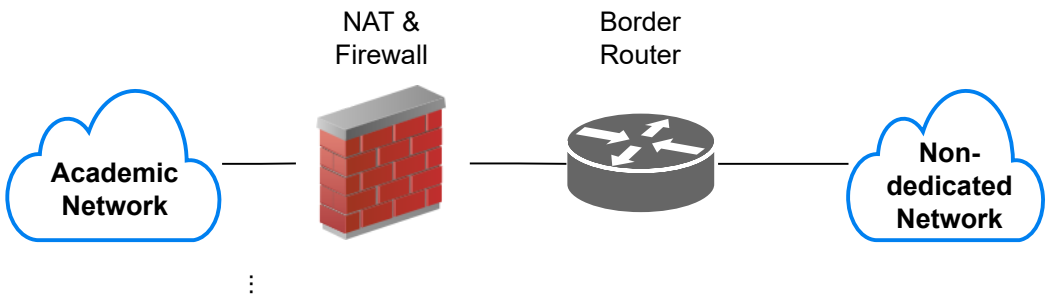



**Figure** 5.1: NATed Academic Network

In this scenario, each flow $f_i$, $i = 1, ..., N$ between the academic and non-dedicated networks can be represented with the 3-tuple {external IP, campus IP, campus port}.

Given $X$ a random variable, its entropy is computed as follows:

$$H(X) = \sum_{i=1}^{N} p(x_i) \log_2 \left( \frac{1}{p(x_i)} \right), \tag{5.1}$$

Where $x_1, x_2, ...x_N$ is the range of values for $X$, and $p(x_i)$ is the probability that $X$ takes the value $x_i$. Entropy measures the randomness of a data set. The more random the data is, the more entropy it contains. Among all probability distributions, the largest entropy corresponds to the uniform distribution: $\log_2(N_0)$. $N_0$ is the number of distinct $x_i$ values present in a time slot.

Now, let $F$, $U$, $V$, $W$ random variables representing the 3-tuple {external IP, campus IP, campus port}, external IP, campus IP, and campus port, respectively, their associated entropies on a given time slot $\Delta T$ are $H(F)$, $H(U)$, $H(V)$, $H(W)$.

Based on the above computation, we aim to characterize the network flows to find patterns that allow us to distinguish between normal and suspicious traffic.

## 5.2 Materials and Methods

### 5.2.1 Topology

Based on the Problem Statement reference topology of Figure 5.1, Figure 5.2 shows the flow monitoring architecture used in this work. Here the academic network is named campus network and the non-dedicated network is Internet. The border router connects the campus network to the Internet Service Provider (ISP) / Internet. The NAT device translates private IP addresses to a single public IP address (campus IP). The campus network corresponds to the Northern New Mexico College (NNMC). It connects 15 buildings and different departments. There are approximately 250 faculty/staff members, 1,500 students, 20 general-purpose computer laboratories, and faculty and staff offices. Many students access the Internet via WiFi using personal devices.

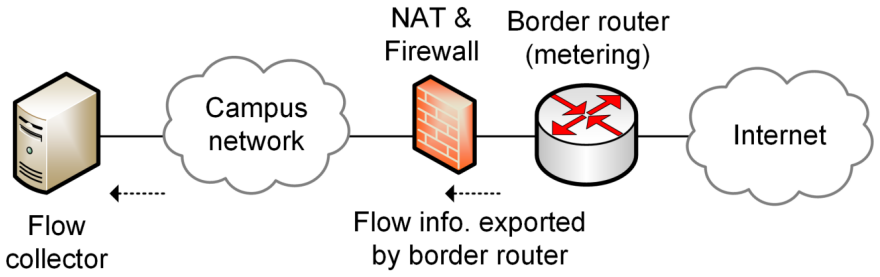



**Figure** 5.2: Topology configuration.

The analysis presented is general and can use multiple metering points and flow directions. For simplicity of implementation in the small/medium-sized network considered here, the metering point is the single border router and the traffic direction is inbound (from the Internet to the campus network). Large campus networks with several border routers may require each border router to become a metering point and to monitor inbound and outbound traffic directions. The border router is a Cisco ASR 1000 series. Flow information is ready for export when i) it is inactive for a certain time (i.e., no new packets received for the flow during the last 15 seconds); ii) the flow is long lived (active) and its duration is greater than the active timer (1 minute); and when a TCP flag indicates that the flow is terminated (i.e., FIN, RST flag are received). For each flow, the router exports the source and destination IP addresses, source and destination

ports, layer-4 protocol, TCP flags observed during the connection, and connection statistics including number of packets, number of bytes, bytes per packet, and flow duration. The information is collected by the flow collector, which implements NetFlow protocol version 9 [150]. To avoid confusion, instead of using source and destination terminology, this paper will use external and campus IP addresses, and external and campus ports. Note that from the point of view of the border router, users' flows have the same campus IP address (public IP), because of NAT. The collector organizes flow data in five-minute time slots. Data analysis is conducted for each individual time slot.

While traffic data has been collected for more than a year, the paper presents the analysis of a typical week, from Saturday, March 25, 2017 to Friday, March 31, 2017. The traffic data observed during this week is representative of the campus traffic.

### 5.2.2 Entropy Measures

We compute entropies as described in [117], [119]. However, this work uses flow counts rather than packet counts.

Therefore we define the following probabilities based on network traces captured in the flow collector:

For each external IP address $u_i$, the probability $p(u_i)$ is calculated as

$$p(u_i) = \frac{\text{Flows with } u_i \text{ as external IP addr.}}{\text{Total number of flows}}. \tag{5.2}$$

For each campus port $w_i$, the probability $p(w_i)$ is calculated as

$$p(w_i) = \frac{\text{Flows with } w_i \text{ as campus port}}{\text{Total number of flows}}. \tag{5.3}$$

The normalization factor is $\log_2(N_0)$, where $N_0$ is the number of active external (campus) IP addresses (ports) observed during the time slot.

In addition, we consider the entropy of the 3-tuple {external IP, campus IP, campus port}. For a given 3-tuple $f_i$, the corresponding probability is calculated as:

$$p(f_i) = \frac{\text{Flows with } f_i \text{ as 3-tuple}}{\text{Total number of flows}}. \tag{5.4}$$

The normalization factor is $\log_2(N_0)$, where $N_0$ is the number of active flows during the time slot.

### 5.2.3 Time-series Correlation and Data Statistics

For each time slot, the five normalized entropies are computed. Let $Y_{i,j}$ denote the normalized entropy of distribution $i$ (e.g., campus IP address) observed in time slot $j$, and $Y_i$ denote the time-series of normalized entropy values for distribution $i$. Given the $Y_i s$, the pairwise correlation coefficients between every pair of time-series vectors $Y_i$ and $Y_{i'}$ are computed [117]:

$$r_{i,i'} = \frac{\sum_j Y_{i,j} Y_{i',j} - n\overline{Y_i Y_{i'}}}{(n-1)\sigma_{Y_i}\sigma_{Y_{i'}}}, \tag{5.5}$$

where $\overline{Y_i}$ and $\overline{Y_i'}$ are the sample means of $Y_i$ and $Y_{i'}$, $\sigma_{Y_i}$ and $\sigma_{Y_{i'}}$ are the sample standard deviations of $Y_i$ and $Y_{i'}$, and $n$ is the number of time slots.

Additional data statistics are also computed: mean, standard deviation, maximum and minimum values. As the measured data sets from weekdays and weekend substantially differ in volume and entropy [151], the data statistics are separately computed for weekend and weekdays,

### 5.2.4 Time-series Rate of Change

The rate of change of the entropies is also approximated as an indicator of anomalies. A simple approximation of the derivative of $Y_i$ with respect to time is computed as the difference between consecutive time slots $j$ and $j+1$:

$$\Delta Y_{i,j} = Y_{i,j+1} - Y_{i,j}. \tag{5.6}$$

## 5.3 Results and Evaluation

Figure 5.3 shows the total traffic and entropy quantities during a typical week. The red portion of the curve represents the weekend (Saturday March 25, 2017 - Sunday March 26, 2017) and the green portion of the curve represents the weekdays (Monday March 27, 2017 - Friday March 31, 2017). The corresponding statistics for the graphs are listed in Table 5.1. The mean traffic volume on the weekday/weekend is $1,101/168$ Mbytes in a 5-minute time slot. Thus, there is a difference of an order of magnitude between weekday and weekend. The day and night patterns are clear on weekdays, when users (students, faculty, and staff) are on campus. The peak time is slightly after 12:00 noon.

Consider the campus IP's entropy. As a small/medium-sized campus network, the number of public IP addresses is limited (less than 50 active IP addresses. One public IP address is used for NAT (users' flows) and few others are used for externally available servers). When users are on campus during weekdays, the

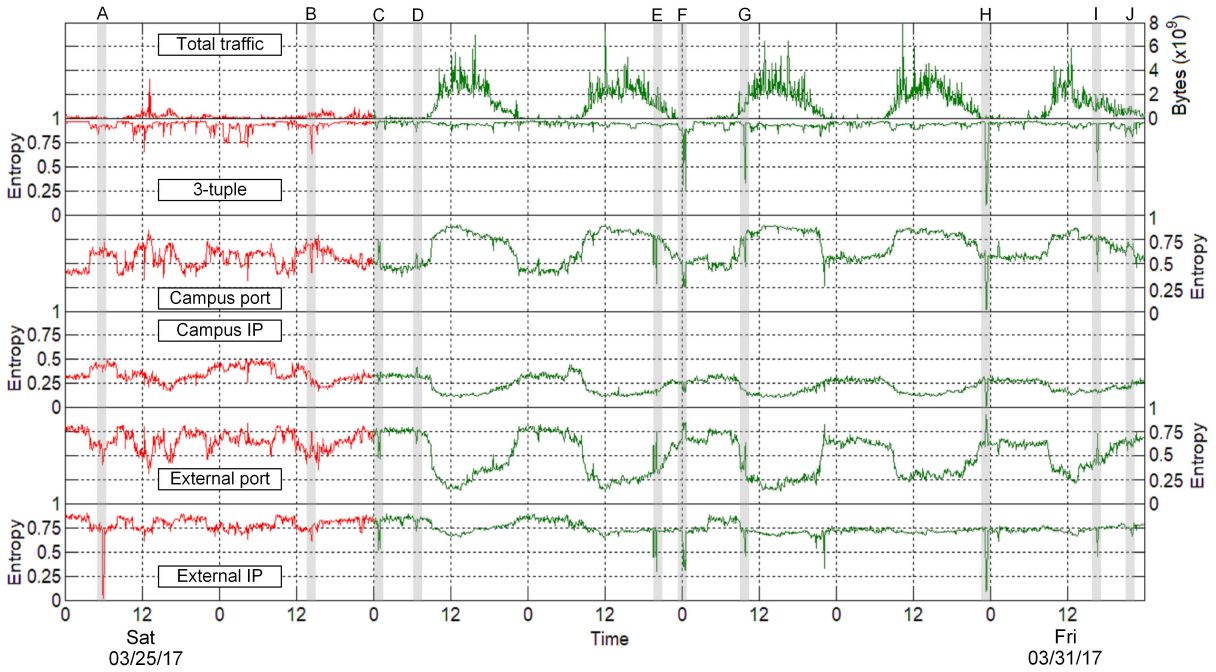



**Figure** 5.3: Entropy time-series for the small/medium-sized network studied in this paper. Anomalies are labeled with letters $A$ through $J$.

**Table** 5.1: Statistical information for the week data of Fig. 5.3.

| Feature | Mean ($\mu$) | Std ($\sigma$) | Max | Min |
|---|---|---|---|---|
| Total traffic weekday / weekend ($x10^6$ bytes) | 1,101 / 168 | 1,254 / 224 | 8,875 / 3,262 | 5.5 / 4.1 |
| 3-tuple entropy weekday/weekend (bits) | 0.938 / 0.933 | 0.061 / 0.049 | 0.98 / 0.979 | 0.104 / 0.632 |
| Campus IP entropy weekday/weekend (bits) | 0.224 / 0.345 | 0.077 / 0.0769 | 0.44 / 0.503 | 0.011 / 0.172 |
| Campus port entropy weekday/weekend (bits) | 0.667 / 0.548 | 0.156 / 0.100 | 0.893 / 0.839 | 0.015 / 0.319 |
| External IP entropy weekday/weekend (bits) | 0.739 / 0.791 | 0.070 / 0.071 | 0.898 / 0.902 | 0.085 / 0.016 |
| External port entropy weekday/weekend (bits) | 0.486 / 0.656 | 0.204 / 0.098 | 0.926 / 0.836 | 0.138 / 0.309 |

entropy can be as low as ∼0.1. The reason of the low entropy is the use of NAT, which maps users' private IP addresses to a single public IP address. During the weekday/weekend, the mean entropy is 0.224/0.345. However, note the large variation. On weekdays, the range $\mu \pm \sigma$ is $0.301 - 0.147$.

The entropy of the campus port diverges from that of the campus IP. During the day, as users connect to the network, the most popular application is browsing. Browsers use ephemeral port numbers, behaving more randomly. At peak hour, the distribution of the campus port approaches a uniform distribution and the entropy approaches ∼0.9. The variation is very large during both weekdays and weekend; e.g., on weekdays, the range $\mu \pm \sigma$ is $0.823 - 0.511$.

Consider the entropy of the external IP address. The entropy is much larger than that of the campus IP. This is a reflection of users connecting to a large

number of websites. However, note that the distribution is far from uniform, as entropy is well below 1. This indicates that there are users connecting to the same sites/IP addresses (e.g., popular sites include YouTube, Google, Facebook). However, note that while the entropy variation is much lower than that of other flow elements discussed above, the range $\mu \pm \sigma$ is $0.809 - 0.669$, still significant.

The external port's entropy shows the largest variation among all distributions. During weekdays, at peak times, the entropy decreases to the lowest value, even below 0.2 some days (Monday, Tuesday, and Wednesday). The users' main application is browsing, thus they connect to few well-known ports (i.e., port 80, 443) which decreases the entropy. On the other hand, the distribution changes in opposite direction around midnight, when the entropy increases to the largest value, $\sim$0.8. Note the large variation, in particular for weekdays. The range $\mu \pm \sigma$ is $0.69 - 0.282$ on weekdays.

The entropy of the 3-tuple {external IP, campus IP, campus port} is the most consistent over time with a distribution that resembles a uniform distribution. Most flows generated by users have a unique 3-tuple. Thus, the entropy is high during both weekdays and weekend, with mean values 0.938 and 0.933 respectively. Note also the low variation. In a network without anomalies, the number of flows having the same 3-tuple would be close to zero. Thus, a deviation from this situation may indicate anomalies.

### 5.3.1 Event Use Cases

Entropy values should be interpreted in a day/time context. For example, a value of 0.25 for the entropy of the external port is not abnormal for a weekday at noon; however, it does represent an anomaly if that value is seen at 6 AM. Along these lines, events $A$ through $J$ represent anomalies. Most of them deviate from the normal values that should be observed at a given day/time.

Before describing few event examples, consider Fig. 5.4(a). External and internal ports are shown in orange and green respectively. Under normal circumstances, most flows are unicast connections with unique 2-tuple {external IP, external port}, unique campus port (if necessary, the NAT device performs port address translation), and common campus IP (NAT public IP address shared by campus users).

#### Event A

This is a SYN flood event from a single perpetrator, thus the external IP's entropy is $\sim$0. While not as pronounced, the external port's entropy also decreases, because the perpetrator uses a relatively small number of ports in relation to the number of flows being generated (875 different ports were observed). Most external ports are used to attempt opening up to $\sim$1,000 connections each. Campus

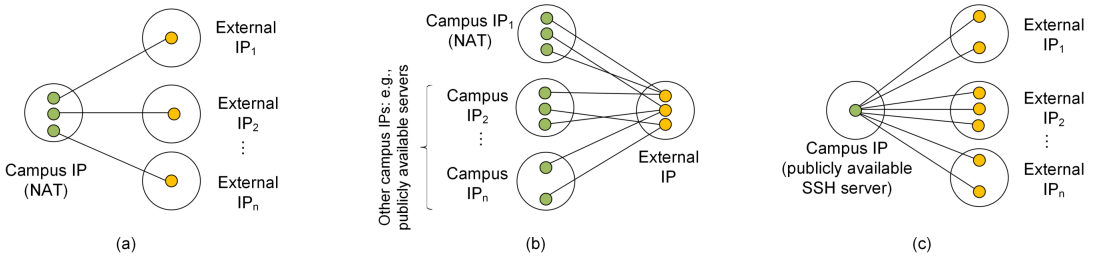



**Figure** 5.4: Flow patterns

IP's and port's entropies do not change, because the attack targeted the entire set of campus IP addresses of the target institution (note that the campus NAT address is one of the IP addresses of the target. Other addresses are allocated to few servers providing specific services, including email and web services) (∼1,000 attempted connections per campus IP address and ∼230 attempted connections per campus port). The 3-tuple's entropy does not indicate anomalies because the volume of any aggregate traffic changes proportionally to the total traffic. On average, there are ∼1,080 flows from the single external IP to each campus IP. Fig. 5.4(b) shows a simplified illustration of the flow pattern observed during event $A$.

### Event B

This event does not correspond to an attack but to DNS activity. Approximately ∼50 Amazon servers generate flows addressed to the local DNS server on campus. Each Amazon server generates between ∼100 and ∼300 flows to the local DNS server. Flows from a single Amazon server have identical 3-tuple {external IP, campus IP, campus port}. Thus, the 3-tuple's entropy is the best event indicator. The campus port's entropy decreases because of the increase in port 53 (DNS) activity. Under normal conditions, the number of DNS flows in a 5-minute window is typically below 1,000. During event $B$, the number of DNS flows increased to more than 6,000. The total traffic in bytes is not an indicator for this event, as the increase in DNS traffic is minimal when compared to the total traffic observed during the time slot.

### Event G

This event occurred at 9:50 AM on Wednesday 03/29/17 and was a dictionary/brute-force attack to an SSH server (campus port 22). The total traffic does not reflect an anomaly, because SSH brute-force login attempts do not produce much volume (the traffic volume in bytes from the perpetrator was below 0.3% of the total traffic volume). However, the anomaly is captured by a drop in the 3-tuple entropy

from ∼0.92 to ∼0.33. While a smaller change is also observed in the campus port's entropy from ∼0.75 to ∼0.5, this change occurs in an opposite direction to the natural entropy change for that day and time of the week (i.e., the normal behavior of the campus port's entropy during a weekday should show a steady increase until approximately noon). Similarly, although the natural tendency during this time slot is the decrease in external port's entropy, the anomaly sharply reverses this trend by increasing the entropy from ∼0.35 to ∼0.6. The increase in the external port's entropy occurs because the perpetrator's device opens several connections using ephemeral ports. A reader can carefully note that an external port's entropy value of ∼0.6 is a valid value for a different time window, but not for the time slot of event $G$. The external IP's entropy also captures the anomaly with a decrease in entropy from ∼0.75 to ∼0.5.

#### Event H

From few external IP addresses, the perpetrators of event $H$ opened multiple connections (using different external ports) to attempt to gain access to a single IP / port on campus. Event $H$ is similar to an SSH dictionary / brute-force attack, but perpetrated by several devices (e.g., botnet). The external IP's entropy decreases as the number of flows from the perpetrators increases. Fig. 5.4(c) illustrates this attack.

Other events labeled as $C, D, E, F, I, J$ show similarities to those described above. Anomalies can be detected by the rapid change in one or more entropy measures.

### 5.3.2 Entropy Time-series Correlation

Table 5.2 shows the correlation between the entropy time-series.

#### Total traffic

During weekdays, the total traffic is negatively correlated to the entropies of the campus IP (-0.8) and external port (-0.81). Traffic increases as a result of users accessing mostly web applications; thus the campus IP's entropy decreases because users use the same campus IP address (NAT public IP). The external port's entropy also decreases because most traffic uses http/https. The total traffic is directly correlated to the campus port's entropy (0.78), because as users on campus access the web, their respective browsers open ephemeral ports that collectively resemble a uniform distribution. On weekend, there is a low or no correlation between the total traffic and other time-series.

**Table** 5.2: Correlation of entropy time-series.

| | Campus IP | Campus port | External IP | External port | Total traffic |
|---|---|---|---|---|---|
| Weekday | | | | | |
| 3-tuple | 0.23 | 0.1 | 0.6 | -0.02 | -0.05 |
| Campus IP | | -0.85 | 0.6 | 0.89 | -0.8 |
| Campus port | | | -0.37 | -0.98 | 0.78 |
| External IP | | | | 0.45 | -0.36 |
| External port | | | | | -0.81 |
| Weekend | | | | | |
| 3-tuple | -0.23 | -0.12 | 0.56 | 0.06 | -0.03 |
| Campus IP | | 0.15 | -0.38 | 0.06 | -0.38 |
| Campus port | | | -0.48 | -0.93 | 0.31 |
| External IP | | | | 0.48 | -0.05 |
| External port | | | | | -0.39 |

### Campus IP

On weekdays, the entropies of the campus IP and campus port are negatively correlated (-0.85). As users use the network, the campus IP's entropy decreases because users use the same campus IP address (NAT public IP). On the other hand, the campus port's entropy increases because users' browsers open ephemeral ports. The entropies of the campus IP and external port show a direct correlation (0.89): the more traffic is generated by users, the more NATed flows exist, and the lower the campus IP's entropy is. As most traffic is http/https, the external port's entropy also decreases. On weekend, there is a low or no correlation between campus IP and other time-series.

### Campus port

On weekdays, the strongest negative correlation is between the entropies of the campus port and external port (-0.98). This relation is produced by the use of a large number of browser's ephemeral ports (each user's browser likely uses a different port number) to connect to few external ports (i.e., http/https). On weekend, there is a low or no correlation between campus port and most distributions, with the exception of external port.

#### External IP

The entropies of the external IP and external port are correlated on weekdays (0.45) and on weekend (0.48). Note that the external IP has high entropy (mean is 0.739) when compared to other flow elements. This reflects the variety of external IP addresses users connect to.

#### External port

As mentioned above, the entropies of the external port and campus IP are strongly correlated (0.89). On the other hand, the entropies of the external port and campus port are negatively correlated on weekdays (-0.98) and on weekend (-0.93).

#### 3-tuple {external IP, campus IP, campus port}

The entropy of the 3-tuple shows correlation with that of the external IP on weekdays (0.6) and on weekend (0.56). Low or no correlation is noted between the 3-tuple and other time-series.

### 5.3.3 Time-series Rate of Change

**Table** 5.3: Statistical information, rate of change of entropy time-series.

| Feature | Mean ($\mu$) | Std ($\sigma$) |
|---|---|---|
| Total traffic weekday / weekend ($x10^6$ bytes/time unit) | 0.2 / 0.18 | 774 / 256 |
| 3-tuple entropy change weekday/weekend (bits/time unit) | ∼0 / ∼0 | 0.057 / 0.038 |
| Campus IP entropy change weekday/weekend (bits/time unit) | ∼0 / ∼0 | 0.021 / 0.03 |
| Campus port entropy change weekday/weekend (bits/time unit) | ∼0 / ∼0 | 0.045 / 0.058 |
| External IP entropy change weekday/weekend (bits/time unit) | ∼0 / ∼0 | 0.04 / 0.05 |
| External port entropy change weekday/weekend (bits/time unit) | ∼0 / ∼0 | 0.04 / 0.056 |

Fig. 5.5 shows the rate of change of the total traffic and entropy time-series, computed according to Eq. (5.6). Corresponding values are provided in Table 5.3. The first observation here is the large rate changes in the total traffic, in particular during weekdays. Thus, traffic rate changes may not always be accurate indicators of anomalies, as they occur naturally in this small/medium-sized network. In contrast, changes in the entropy time-series from one time-slot to another (in bits / time unit) are small. The mean values for entropy changes are approximately zero.

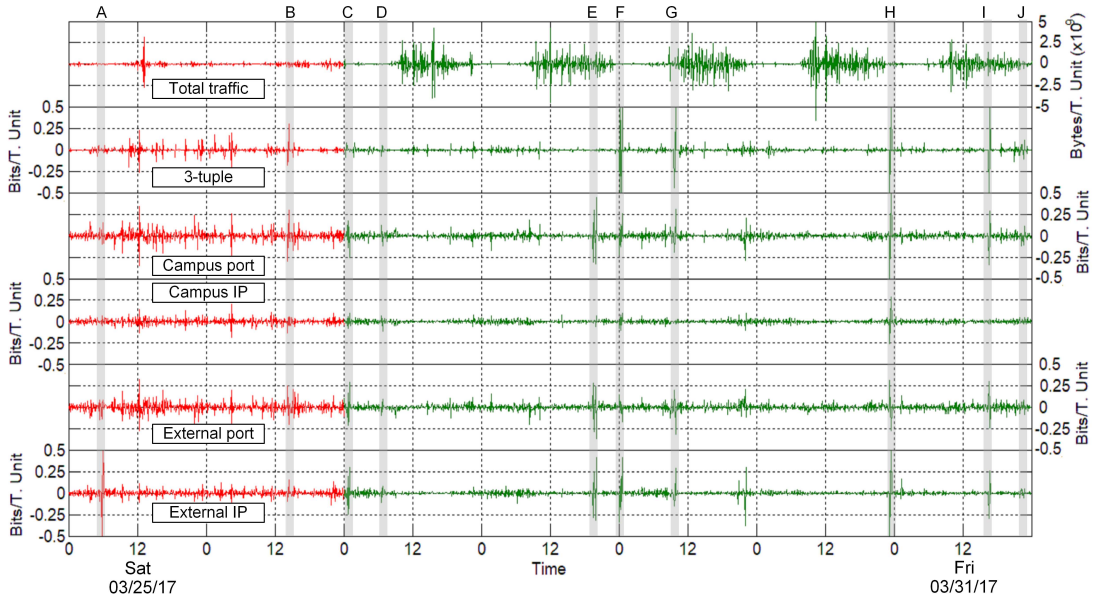



**Figure** 5.5: Rate of change for the time-series shown in Fig. 5.3. Values are computed using Eq. (5.6).

## 5.4 Conclusion

This chapter presented a flow-based entropy characterization of a small/medium-sized academic network that uses NAT. Measurements from a production network show that on a typical weekday, the entropies of the external and campus ports may widely vary from below 0.2 to above 0.8 (in a normalized entropy scale of 0-1). Similarly, the entropy of the campus IP address may vary from 0.1 to 0.4. Despite the wide range of values, findings indicate that building a granular (small time slots) entropy characterization of flow elements facilitates anomaly detection. Data shows that specific attacks produce entropies that deviate from the expected patterns. Data also shows that the entropy of the 3-tuple {external IP, campus IP, campus port} is high and consistent over time, resembling the entropy of a uniform distribution's variable. A deviation from this pattern is an encouraging anomaly indicator.

The above findings remark that the entropy approach could help detect anomalies at high rates. Hence, it can be useful in scenarios where a friction-free path is desired, such as Science DMZ, because collecting the patterns, calculating the entropies, and detecting the anomaly can be done parallel to the production network. As in the studied case of congestion control, data plane programmable devices are potential technologies to perform such procedures.

# 6 Framework for SDN Teaching and Research

Considering that one research aims to strengthen the learning process, we propose a framework that leverages the close relationship between the academy, real-world ICT problems, and innovative service. The framework is intended to be extended to SDN, considering it is an agile and continuous development technology.

The Chapter is organized as follows. Section 6.1 describes the main motivations around establishing a framework for research and teaching in SDN and this relationship with the devolpment of Science DMZ. Section 6.2 details the proposed framework emphasizing the relationships between components, actors, and stakeholders. The appreciation perceived by the students and alumni is addressed in Section 6.3. Finally, Section 6.5 presents the relevant conclusions.

## 6.1 Motivation

Software-Defined Networking (SDN) technology traces its origins to a research laboratory at Stanford University and has emerged as one of the most crucial networking paradigms in the last decade. The key innovation with SDN has been to decouple the data and control planes, allowing operators to directly manage network resources and adapt to customer needs through programmable applications. As such, this paradigm supports agile and flexible services for a wide range of users. Overall, SDN continues to gain attention and many of its technological solutions have been accepted and applied in the Information and Communication Technology (ICT) industry, showcasing the applicability of research and development efforts in the field. Indeed, this new perspective has enabled solutions for both private enterprise domains as well as large-scale public cloud infrastructures. Furthermore, SDN is also emerging as a critical component in upcoming 5G core networks by providing an intelligent, flexible, and programmable architecture [152].

SDN is also an innovation driver to improve the Science DMZ. There are several key applications of SDN on Science DMZ as follows:

- *Network Optimization*: SDN can be used to optimize network traffic flows within Science DMZ by dynamically configuring network devices based on real-time data flow requirements [153]. This helps to reduce network congestion and latency, and ensures that data is transferred quickly and efficiently.
- *Traffic Prioritization*: SDN can prioritize certain types of traffic, such as real-time data, over other types of traffic in Science DMZ networks [154]. This helps to ensure that critical data is given priority and delivered in a timely manner, even during periods of heavy network traffic.
- *Security*: It has been demonstrated that SDN can help to improve network security in Science DMZ by enabling administrators to implement granular access controls and security policies for different types of traffic [155, 156, 157, 158] . This helps to reduce the risk of data breaches and cyber-attacks.
- *Flexibility*: SDN enables administrators to easily reconfigure network devices in Science DMZs to accommodate changes in data flow patterns or research requirements. This helps to ensure that the network can adapt quickly to evolving research needs.
- *Resource Allocation*: SDN can be used to allocate network resources, such as bandwidth and processing power, to different data flows in Science DMZs based on their priority and importance. This helps to ensure that critical data flows receive the necessary resources to support efficient and timely data transfer.

Energy Science Network (ESnet) proposes that a sub-network can be created within the Science DMZ for experimental purposes to incorporate SDN functions and enhancements into the production network gradually [8], as shown in Figure 6.1.

Given the variety of expectations, requirements, and constraints of networking users, SDN-based solutions require skillful and proficient practitioners to adequately address key analysis, design, implementation, and operational requirements. As a result, there is a growing need to include SDN-related knowledge and skills in undergraduate networking course curricula [159, 160]. However, teaching such concepts poses a range of challenges as compared to other engineering curriculum topics. Foremost, SDN requires key networking-related skills (design and operation) as well as code development background. This domain also requires up-to-date training infrastructures and related educational materials to develop meaningful, practical experiences. Finally, commercial tools and methodologies

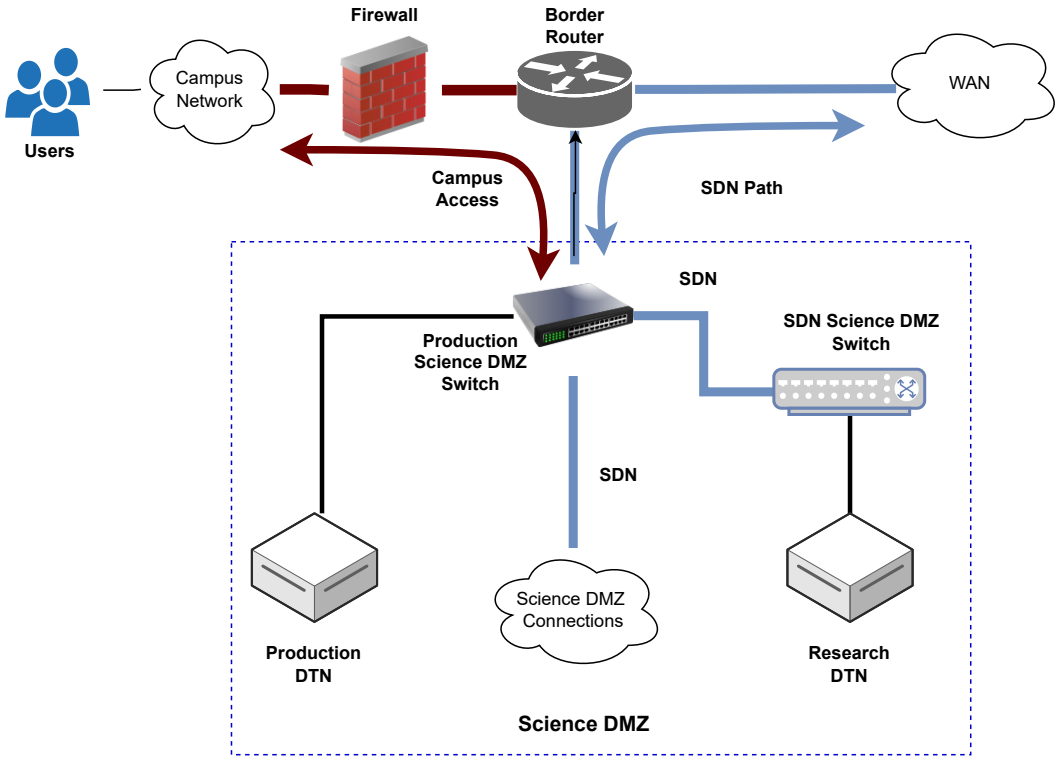



**Figure** 6.1: Science DMZ for Software Defined Networks

in the SDN sector are continually evolving and have relatively short life cycles; related courses can easily become dated or obsolete [123].

## 6.2 Framework for SDN Teaching and Research

Given the complexity of SDN-related teaching-learning and research processes, a holistic framework was developed to leverage and understand the existing relationship between key actors and stakeholders. In the proposed framework, all the components must generate products and keep the knowledge base updated to sustain the training enterprise over time. Figure 6.2 depicts our approach.

### 6.2.1 Actors and Stakeholders

Foremost, the higher education institution is considered as the principal actor in the knowledge generation process and is represented by professors, researchers, and ICT professionals. Accordingly, our framework promotes robust interactions

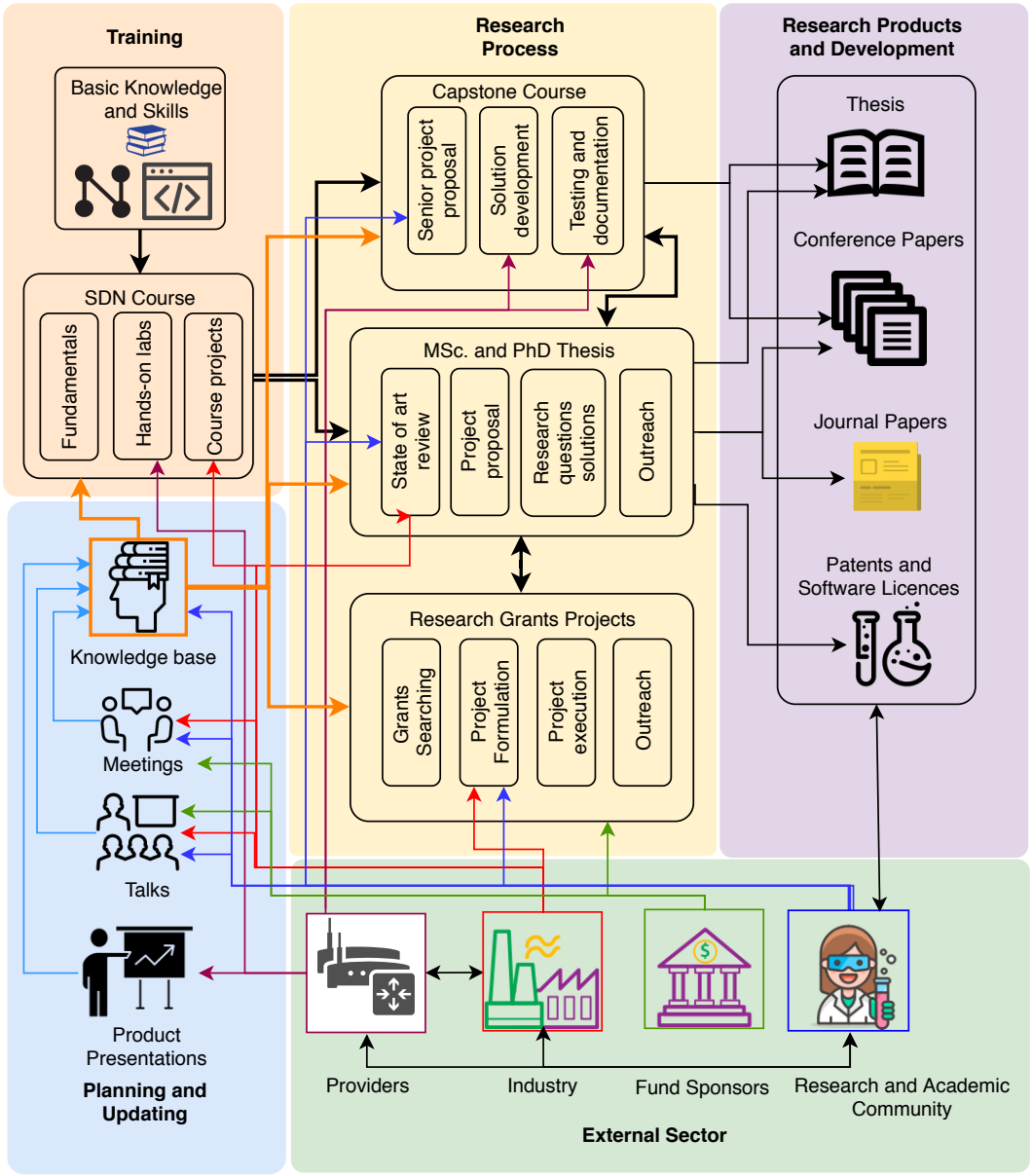



**Figure** 6.2: Interactions between components, actors, and stakeholders in the proposed SDN teaching and research framework

to help group dynamics, induce collaboration, and set the stage for achieving common objectives. In this context, a researcher is any individual involved in the knowledge generation process and contributing to project goals. As such, this group includes undergraduate, Masters, and doctoral students as well as profes-

sors and visiting scholars collaborating on SDN-related projects. In particular, the undergraduate Telecommunications Engineering program at Universidad de Concepción (UDEC) emphasizes a problem-solution based methodology which imparts active practical skill sets for undergraduates. Furthermore, the ICT professional staff at UDEC is also included as it can also perform network development tasks.

The academic and research communities play a leading role in driving the direction of applied SDN research, validating developed products/prototypes, and generating up-to-date knowledge to support the teaching process. External organizations provide financial support and turn research outcomes into products. More importantly, sponsors align research and training efforts with broader regional and national needs and development plans. Many industry organizations are quite interested in applying SDN technologies in their infrastructures and operations. Hence the solutions proposed in this framework are aimed at specific problems identified by such industry partners. Providers are ICT companies granting access to programmable SDN devices.

### 6.2.2 Phases and Components

The proposed framework uses *components* to outline and reduce the complexity of the SDN teaching-learning and research process. Components are defined as those sets of activities aimed at achieving an objective. As shown in Figure 6.2, components must interact with each other to generate value.

In the *Training phase*, the **Basic Knowledge and Skills** component represents the set of courses developing soft skills in the students. Examples of such skills are analyzing real-world ICT problems, interpreting texts, and communicating ideas in oral and written forms. This component also requires that students integrate knowledge on logic, statistics, programming, and networking fundamentals. These skill sets are considered essential for solving the technical challenges presented in assignments, course projects, and research works. Overall, the related training is acquired through curriculum courses and also work/internship experiences.

A fundamental element in this phase is the **SDN Course**. This new laboratory-based course on SDN technology was designed and introduced in the curriculum. Following Bloom's taxonomy, its learning outcomes are: (i) Explore fundamental SDN concepts; (ii) Demonstrate hands-on knowledge on SDN networks; (iii) Design network solutions based on SDN technology; and (iv) Evaluate the performance of SDN networks as compared to traditional solutions. Hands-on experiments are carried out using emulators and actual SDN switches and controllers. The course was designed to encourage critical thinking, problem solving skills, research skills, and team-work. The course employs three didactic methods: lec-

tures, hands-on laboratories, and course projects. Course topics include SDN architecture and protocols, data measurement tools, flow control design and programming, and applications in security and fault tolerance. Professors minimize the number of lectures and focus on supervise student's work.

Course projects follow a Challenge-Based Learning (CBL) approach that encourages solving real-world problems [161]. Students must propose projects related to SDN deployments or a recurrent ICT industry problem that can benefit from an innovative SDN solution. Group meetings are then held to identify the key challenges and problems to be solved. Subsequently, working groups are formed to develop a particular solution, thereby strengthening planning, analysis, design, and implementation skills. Students must also manage their time and projects are peer-reviewed by them, thus emulating discussions among colleagues in a working environment.

The *Research Process phase* focuses on generating knowledge, from the perspective and needs of industry, academia, and government, through the following elements:

**Capstone Course:** This course is an individual undergraduate thesis, with a workload of approximately 700 hours, aiming to solve a guided Capstone Project. As stated by CBL, students must propose a senior project for solving a practical problem applying the theoretical knowledge and skill sets acquired through the curriculum. (Note that Chilean undergraduate engineering programs span six years, providing students with ample time to partner with companies to identify relevant problems.) The addition of SDN-related coursework and training opens up a wide range of potential topics in the ICT space. Students must submit a full report summarizing their objectives and technical achievements. Depending upon results, some students are encouraged to submit their works as scientific conference papers. Papers boost interest in applied research, at the undergraduate level, and also build a reliable pathway to pursue graduate studies.

We highlight that the Capstone Course has been used not only to develop technical SDN solutions but also for educational purposes. In fact, all the laboratory activities were developed by a student that was challenged to design new laboratory material for a hands-on learning experience in SDN. Such a remarkable experience was replicated to create laboratory documentation for Software-defined Radios and Wireless Communications courses.

**Research Grants:** Funding plays a vital role in supporting research. Moreover, research grants can help drive outputs to address significant problems and contribute to broader industrial and societal needs. Along these lines, there are two primary sources of academic funding in Chile, both of whom are actively supporting undergraduate research projects at UDEC. In particular, the National Commission for Scientific and Technological Research (CONICYT) focuses on cutting-edge basic and applied research, whereas the Corporation for the Produc-

tion Promotion (CORFO) supports more comprehensive Research, Development, and Entrepreneurship (RDE) initiatives. In many cases, graduate students are also involved in grant writing initiatives, providing invaluable academic training. **Graduate Programs:** UDEC offers both Masters (M.Sc.) and doctoral (Ph.D.) degree programs in Electrical Engineering. These framework components represent structured processes with high methodological and scientific rigor where graduate students must solve problems and create new knowledge. The key focus at the Masters level is to expose and train students in acquiring a research methodology. Notably, graduation requirements at UDEC consider submitting a journal paper. Meanwhile, the doctoral degree emphasizes originality and knowledge impact at a broader global level. Doctoral graduation requirements include publishing one and submitting another journal paper. However, paper requirements can be exchanged by patent applications and grants, thereby encouraging RDE technology transfer.

Finally, the *Research Products and Development phase* is the last block in the framework. Indeed, a large part of our efforts focus on this phase, as it plays a critical role in disseminating the generated knowledge to broader scientific and industrial communities. Foremost, masters and doctoral theses generate conference and journal papers, some patents, and software licenses in the SDN domain. As noted above, the Capstone Course also mandates a thesis covering the problem space, solution design process, and outcomes. Professors, researchers, and qualified ICT staff carefully review these documents to identify works exhibiting a high degree of originality, coherence, and real-world ICT applicability, for further publication.

### 6.2.3 Feedback in the Framework

As detailed earlier, our framework defines several phases with a logical progression sequence to build sustainable SDN-related teaching and research processes, i.e., from training, to research, to research-product development. However, an effective program must incorporate constant feedback from the lessons learned to update the knowledge base. The framework includes some critical planning and updating components. Foremost, due to continuous dynamics, the theoretical and practical elements of the SDN course are reviewed yearly. The review process accounts for advances in technology and standards, SDN equipment supplier product cycles, and the needs of the scientific and academic communities. As a result, the course syllabus is updated at the beginning of the year based upon meetings between faculties, former students, and alumni.

The components of the research phase are updated likewise. For instance, the Capstone Course must take into account students' existing knowledge base and aim to tackle current, relevant problems in SDN deployment using the latest tools

**Table** 6.1: Summarized survey statements and average grades.

| Statement | Avg. grade |
|---|---|
| Do data networking and system management skills developed in the SDN course surpass those in traditional data network courses? | 6.33 |
| Did you plan to develop an SDN Capstone Project? | 6.64 |
| Are course contents studied up to date? | 6.67 |
| Are course contents presented in-depth? | 6.67 |
| Do SDN skills improve professional development? | 8.08 |
| Do SDN laboratories improve networking skills and clarify data network basic concepts? | 8.42 |
| Was the SDN course challenging? | 8.55 |
| Does the SDN course allow students to choose alternative solutions to networking problems? | 9.00 |

and techniques. As a result, regular interactions are held with external researchers, service providers, and equipment suppliers to obtain crucial feedback. Furthermore, paper publication requirements at the graduate level encourage new research contributions. Thus, any proposed research topic must also incorporate key trends from the industrial and academic communities. Once a research project has been completed, a team conducts further self-assessments to identify key strengths and weaknesses as well as an improvement plan. The team includes professors, researchers, and graduate students. Inputs may be exploited by grant submissions to increase funding success rates. Furthermore, assessments are also conducted for unsuccessful grant proposals to identify future areas of improvement.

## 6.3 Former Students and Alumni Feedback

To assess students and alumni satisfaction, we designed a survey with questions about the effectiveness of the SDN and the Capstone courses. The survey was collected on-line from a target population of 43 former students and a response rate of 33% was achieved, a value typically regarded as a really good response rate. The survey contained a total of 14 questions. Out of the 14, 13 questions should be answered using a 10-grade Likert scale. The last item was an open-ended question requesting comment to improve the courses and their learning outcomes. Due to space constraints, we list Table 6.1 8 out of the 14 statements proposed to the survey respondents. The average grades for these eight statements are also listed. We note that statements in Table 6.1 are not listed following the survey

order.

All in all, the average grades show an excellent reception from the former students and alumni to our courses. (Recall that in Likert scale, the grades 1, 5, 6 to 7, 8 to 9, and 10 respectively represent a strong disagreement, undecided, a mild agreement, an agreement, and a strong agreement with a statement.) The perception of the students is that course contents are relatively up to date and presented in-depth. In all honesty, such evaluation was a bit surprising to us, but from the analysis of the open-ended question, we realized that they suggest to include Internet of Things (IoT) and cloud computing to the courses. Our former students mildly agree that data networking and system management are better comprehended using an SDN-based approach. Despite this mild agreement, they do agree that SDN leverages their professional development and networking skill. We note that students clearly realize that SDN technology leads to innovative networking solutions. Besides, they also regard the courses as challenging.

In summary, survey results show that the combination of challenging topics, the use of hands-on tools, and the motivation for creativity and a problem-solving spirit produce in the students a high degree of satisfaction. Lastly, we note that the most relevant comments and recommendations provided by our former students for course improvement are: (1) "SDN laboratory experiences require more network devices that those currently available;" (2) "Course topics need to be updated;" (3) "Laboratory experiences should include IoT and cloud computing applications;" (4) "Professors need to increase the engagement with ICT companies, allowing students to deal with more practical, real-world problems;" and (5) "Professors should spend more time on lectures instead of just supervising students."

## 6.4 Examples of the Research Process Phase Developments

Next, we briefly present some examples of the SDN-related research and development experiences we carried out at UDEC using the proposed framework.

### 6.4.1 Dynamic Resource Management

CORFO funded a joint project between UDEC and Universidad Santa María to prototype a network-provisioning and monitoring system in the National University Network (REUNA) production network, which connects most of the Chilean universities. Hewlett-Packard was the equipment provider and supplied OpenFlow-enabled switches. A team of four undergraduate students of Telecommunications

Engineering developed their Capstone projects providing solutions for specific components of the final prototype.

The OpenFlow application QoSApp was developed to provide Quality of Service (QoS) support for multimedia traffic by dynamically managing network bandwidth [162]. QoSApp periodically measures the RTT between links and classifies network flows as either best-effort or QoS solving an optimal routing problem subject to constraints on transmission delay and packet losses. Figure 6.3 shows the overall architecture of QoSApp which consists of several modules for calculating optimal routes, monitoring network states, and managing network flows using a Floodlight controller. Overall, this effort demonstrates a successful application case of the proposed framework, where undergraduate students applied their knowledge and SDN-related skills in a production network with resources and support from sponsors.

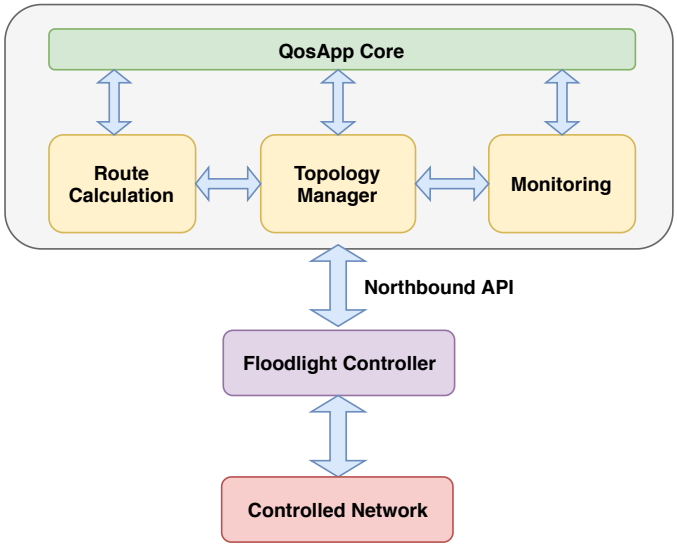



**Figure** 6.3: Architecture of the QoSApp presented in [162]

### 6.4.2 Dynamic Flow Insertion and Handover in WiFi Networks

The ICT staff at UDEC challenged students to implement mobile user handover between WiFi Access Points (APs). A student proposed in his Capstone project to solve this problem by using the artificial flow insertion and measurement capabilities of SDN. He designed and prototyped a laboratory testbed to evaluate the three handover procedures depicted in Figure 6.4.

In the first procedure, termed as Dynamic Flow Insertion, the handover is performed by the SDN controller forwarding the flows between the ethernet interfaces of the APs, which keep track of the roaming users. The second procedure, termed as Proactive Flow Insertion, exchanges data at the OpenFlow switch by sending traffic to both APs. The third procedure, termed as Bi-Casting Reception, exploits the use of two physical wireless interfaces (wlan0 and wlan1) to exchange data from the APs and forwarded to a virtual interface (veth1) using Open vSwitch and OpenFlow at the clients. The assessment of the three solutions showed that the first approach produces handover delays of 1 or more seconds and a Lost Packet Rate (LPR) of 5%. The second approach reduces the handover delay bu increases the LPR, while the third scheme introduces delay but no packet losses.

This project shows how the CBL methodology stimulates the creativity of students and how SDN technology further encourages the development of innovative networking solutions.

### 6.4.3 Reliable Network Provisioning and Survivable Migration in SDN Environments

Disaster recovery is a significant concern in Chile. The National Council of Innovation for Development (CNID) opened a call for research projects to investigate infrastructure resilience in the presence of natural disasters and cyberattacks. CONICYT agency funded a research project to leverage SDN technologies and develop innovative path provisioning strategies for maximizing network reliability under large-scale correlated failures. This project has been carried out by a team of professors and doctoral students from Chile, Colombia, and Cuba. SDN technology offers the inherent flexibility, compatibility, and low-cost administration for designing multi-vendor technology networks. Researchers have exploited these features and developed a novel *multi-culture design* strategy to improve network reliability and resilience. This multi-culture network design approach combines different vendor platforms and operating systems. Next, they formulate optimization problems to determine how many nodes of each vendor are required to minimize the vulnerability of the entire network. Besides, they also optimally determine where such nodes should be located to maximize the average network connectivity. Results in [163] show that, for the Chilean REUNA network topology, placing the more reliable technologies in a clustered manner improves overall network reliability as compared to a single-vendor network. Currently, the team is extending its multi-vendor network design concept to formulate a reliable migration methodology supported by risk-diverse SDN implementations [164]. In line with the feedback requirement in the proposed framework, the knowledge base generated from these particular findings will be further analyzed in future SDN

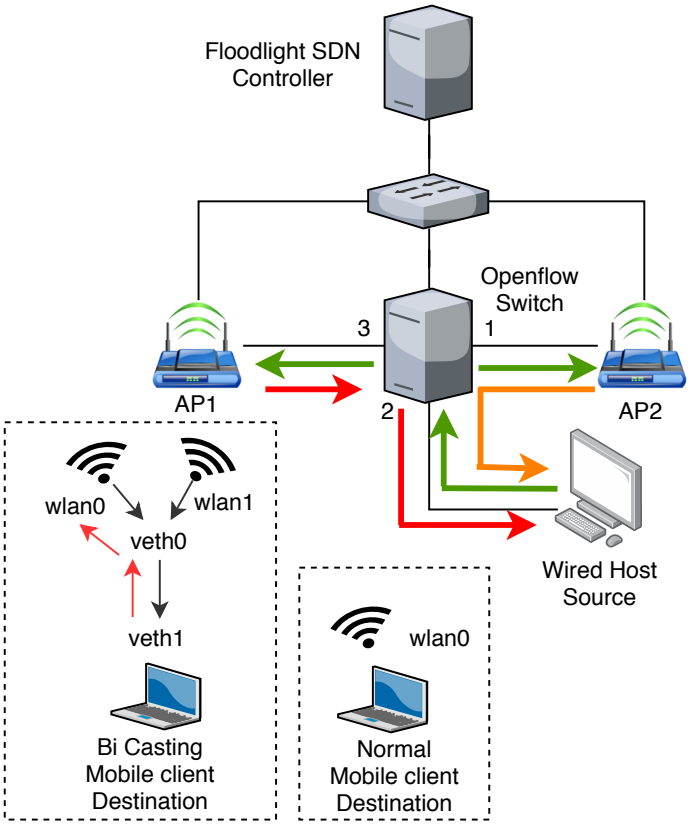



**Figure** 6.4: Testbed for WiFi handover implementation depicting the three approaches developed: Dynamic Flow Insertion, Proactive Flow Insertion, and Bi-Casting Reception.

testbed projects.

## 6.5 Conclusion

This chapter presented a novel framework for SDN-related teaching and research. This framework is defined by a set of actors, stakeholders, phases, and components, as well as their relationships. Overall, its main objective is to build a sustainable domestic knowledge base in this vital ICT field and transition research products to the broader community. As such, the framework is heavily premised upon close working relationships with a range of external partners, e.g., including equipment suppliers, ICT industry corporations, funding sponsors, and the academic and research communities.

The application of the framework has been extremely beneficial for students, helping them build much-needed skill sets in a critical technology sector. Also,

the framework leverages their professional development by exposing them to real-world applied-research problems. In particular, students have been able to expand their existing skill sets in programming, networking, statistical analysis. Also, they acquired additional knowledge through a new SDN course and a range of research activities. Indeed, some notable students outcomes have further validated the effectiveness of this endeavor, as evidenced by the development of products such as papers and software applications. Consequently, this framework provides a robust template which can be adapted and deployed at other institutions to further student training and research in the SDN domain.

# 7

# Conclusions and Future Work

The present work addressed the challenge of enhancing scientific information sharing over non-dedicated networks and contributed solutions and insights that could advance the state-of-the-art in this area. One of the main contributions of this work is the effective integration of data plane devices capabilities and intelligent control techniques using Machine Learning models applied to improve SBD transmission over non-dedicated cyberinfrastructures. Specifically, a novel framework was developed so that from passive measurements in the data plane, relevant control actions are taken to guarantee the transmission of SBD flows and anticipate congestion events. The testbed results demonstrated that the approach increases the Flow-Completion Time (FCT) of SBD with a minimal impact on general purpose traffic. One of the advantages of the proposed solution is that it does not generate overhead in the production network because measurements are taken passively, and control information is carried through the management interface.

Pipeline processing and the limitations of the architecture impose constraints on the design and development of algorithms at the data plane. However, detailed knowledge of the operation of the protocols and the use of hash functions to identify flows, stateful elements, and interfaces with the control plane allowed the development of reliable network telemetry schemes. These measurements allow for determining the state of the network at a given time and are the main input for the control scheme developed.

Another relevant contribution of this work is the development of a rate control based on Machine Learning, which regulates the output rate of the Campus network flow, anticipating congestion events that impact the scientific data flow, thanks to the fact that the controller considers present and past states of the network operation. The proposed controller has a neural network architecture with two hidden layers to prevent considerable computation delays and allow its online

operation on the network.

While the central focus of the present dissertation was congestion control, the problems of bottleneck router's buffer size estimation and anomalous traffic detection were also addressed from the data-driven approach. The results demonstrated the potential of leveraging the information in indirect measurements and flow traces to identify behavioral patterns and predict network operating conditions.

Considering the SDN layer model, several contributions of this work are framed in improvements in the data plane, which are expected to be fed back to the Department of Electrical Engineering courses and projects. In order to motivate long-term study and research on topics related to SDN, a framework to enhance the knowledge base was also addressed in the dissertation.

In terms of future work, there is still much to be done in the field of SBD flows transferring over non-dedicated infrastructures. The work developed considered only TCP as the congestion protocol since it is the most widely used in current non-dedicated networks. The proposed framework could be adapted to no-connection oriented transport layer protocols such as User Datagram Protocol (UDP) and the advent of new CCA for TCP. Although the models developed in the thesis were trained with sufficient data and under various conditions, reinforced learning models can be explored to dynamically adjust the controller parameters under new CCA, chaotic conditions, or network failures. The data-driven solutions for buffer size estimation and anomaly detection developed in this dissertation can be implemented in future work with the support of data plane programmable devices, further enhancing their relevance. Recently, many network functions have been offloaded onto programmable devices in the data plane, and evaluating the feasibility of offloading ML algorithms onto the data plane is a promising field of research. Critical aspects for deploying these solutions, such as scalability and security, deserve to be studied.

# Bibliography

[1] I. Monga, E. Pouyoul, and C. Guok, "Software-defined networking for big-data science-architectural models from campus to the wan," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*. IEEE, 2012, pp. 1629–1635.

[2] T. Niemi, J. K. Nurminen, J.-M. Liukkonen, and A.-P. Hameri, "Towards green big data at cern," *Future Generation Computer Systems*, vol. 81, pp. 103–113, 2018.

[3] Z. Wu, J. Gu, Y. Li, F. Xiao, J. Sun, and Z. Wei, "Distributed parallel endmember extraction of hyperspectral data based on spark," *Scientific Programming*, vol. 2016, p. 2, 2016.

[4] J. Chung, S. Donovan, J. Bezerra, H. Morgan, J. Ibarra, R. Clark, and H. Owen, "Novel network services for supporting big data science research," *Future Generation Computer Systems*, vol. 98, pp. 512–521, 2019.

[5] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science dmz: A network design pattern for data-intensive science," *Scientific Programming*, vol. 22, no. 2, pp. 173–185, 2014.

[6] J. Crichigno, E. Bou-Harb, and N. Ghani, "A comprehensive tutorial on science dmz," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 2041–2078, Secondquarter 2019.

[7] M. Welzl, "Network congestion control," *Proc. of Managing Internet Traffic*, 2005.

[8] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science dmz: A network design pattern for data-intensive science," in *SC '13: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, Nov 2013, pp. 1–10.

[9] L. Smarr, C. Crittenden, T. DeFanti, J. Graham, D. Mishin, R. Moore, P. Papadopoulos, and F. Würthwein, "The pacific research platform: Making high-speed networking a reality for the scientist," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018, pp. 1–8.

[10] W. E. Allcock, B. S. Allen, R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Chard, I. Foster, L. Lacinski, M. E. Papka, and R. Wagner, "Petrel: A programmatically accessible research data service," in *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning)*, 2019, pp. 1–7.

[11] W. Ahmad, B. Alam, S. Sharma, and A. Kushwaha, "Epigenomics scientific big data workflow scheduling for cancer diagnosis in health care using heterogeneous computing environment," *Brazilian Archives of Biology and Technology*, vol. 66, 2022.

[12] E. S. Network, "Esnet fasterdata knowledge base," 1 2023.

[13] B. Tierney, E. Dart, E. Kissel, and E. Adhikarla, "Exploring the bbrv2 congestion control algorithm for use on data transfer nodes," 2021, Conference paper, p. 23 – 30, cited by: 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85124521247&doi=10.1109%2fINDIS54524.2021.00008&partnerID=40&md5=0e70b4be701b84fa29477c1a191adaf2

[14] K. Jutawongcharoen, V. Varavithya, K. Lekdee, A. Chaichit, and T. Sribuddee, "The implementation of the uninet's research dmz," in *2016 International Computer Science and Engineering Conference (ICSEC)*. IEEE, 2016, pp. 1–5.

[15] S. Shah, W. Wu, Q. Lu, L. Zhang, S. Sasidharan, P. DeMar, C. Guok, J. Macauley, E. Pouyoul, J. Kim, and S.-Y. Noh, "Amoebanet: An sdn-enabled network service for big data science," *Journal of Network and Computer Applications*, vol. 119, pp. 70 – 82, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518302236

[16] R. Gegan, B. Perry, D. Ghosal, and M. Bishop, "Insider attack detection for science dmzs using system performance data," in *2020 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2020, pp. 1–9.

[17] H. Li, F. Zhang, L. Yu, J. Oakley, H. Hu, and R. R. Brooks, "Towards efficient traffic monitoring for science dmz with side-channel based traffic winnowing," in *Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*. ACM, 2018, pp. 55–58.

[18] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for sdn," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 99–110, 2013.

[19] G. Gibb, G. Varghese, M. Horowitz, and N. McKeown, "Design principles for packet parsers," in *Architectures for Networking and Communications Systems*. IEEE, 2013, pp. 13–24.

[20] P. Benáček, V. Puš, H. Kubátová, and T. Čejka, "P4-to-vhdl: Automatic generation of high-speed input and output network blocks," *Microprocessors and Microsystems*, vol. 56, pp. 22–33, 2018.

[21] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, "Pisces: A programmable, protocol-independent software switch," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 525–538.

[22] P. B. et. al, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[23] Open Networking Foundation, "P4 Contributors," 2023. [Online]. Available: https://p4.org/ecosystem/

[24] M. Budiu and C. Dodd, "The p416 programming language," *SIGOPS Oper. Syst. Rev.*, vol. 51, no. 1, p. 5–14, sep 2017. [Online]. Available: https://doi.org/10.1145/3139645.3139648

[25] M. Neves, L. Freire, A. Schaeffer-Filho, and M. Barcellos, "Verification of p4 programs in feasible time using assertions," in *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies*, ser. CoNEXT '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 73–85. [Online]. Available: https://doi.org/10.1145/3281411.3281421

[26] E. F. Kfoury, J. Crichigno, and E. Bou-Harb, "An exhaustive survey on p4 programmable data plane switches: Taxonomy, applications, challenges, and future trends," *IEEE Access*, vol. 9, pp. 87 094–87 155, 2021.

[27] J. Zhang, Z. Yao, Y. Tu, and Y. Chen, "A survey of tcp congestion control algorithm," in *2020 IEEE 5th International Conference on Signal and Image Processing (ICSIP)*, 2020, pp. 828–832.

[28] V. Arun, M. Alizadeh, and H. Balakrishnan, "Starvation in end-to-end congestion control," in *Proceedings of the ACM SIGCOMM 2022 Conference*, 2022, pp. 177–192.

[29] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *ACM Queue*, vol. 14, September-October, pp. 20 – 53, 2016. [Online]. Available: http://queue.acm.org/detail.cfm?id=3022184

[30] J. Crichigno, Z. Csibi, E. Bou-Harb, and N. Ghani, "Impact of segment size and parallel streams on tcp bbr," in *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, pp. 1–5.

[31] M. Hock, R. Bless, and M. Zitterbart, "Experimental evaluation of bbr congestion control," in *2017 IEEE 25th International Conference on Network Protocols (ICNP)*, 2017, pp. 1–10.

[32] J. Gomez, E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "A performance evaluation of tcp bbrv2 alpha," in *2020 43rd International Conference on Telecommunications and Signal Processing (TSP)*, 2020, pp. 309–312.

[33] N. Cardwell, Y. Cheng, S. H. Yeganeh, P. Jha, Y. Seung, K. Yang, I. Swett, V. Vasiliev, B. Wu, L. Hsiao *et al.*, "Bbrv2: A model-based congestion control performance optimization," in *Proc. IETF 106th Meeting*, 2019, pp. 1–32.

[34] S. Zhang, W. Lei, W. Zhang, and H. Li, "An evaluation of bottleneck bandwidth and round trip time and its variants," *International Journal of Communication Systems*, vol. 34, 6 2021.

[35] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Transactions on networking*, vol. 1, no. 4, pp. 397–413, 1993.

[36] D. A. Alwahab and S. Laki, "A simulation-based survey of active queue management algorithms," in *Proceedings of the 6th International Conference on Communications and Broadband Networking*, 2018, pp. 71–77.

[37] G. Patil, S. I. McClean, and G. Raina, "Drop tail and red queue management with small buffers: Stability and hopf bifurcation," *ICTACT Journal on Communication Technology*, vol. 02, pp. 339–344, 2011.

[38] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Rfc 8289: Controlled delay active queue management," *IETF, Jan*, 2018.

[39] S. Muhammad, T. J. Chaudhery, and Y. Noh, "Study on performance of aqm schemes over tcp variants in different network environments," *IET Communications*, vol. 15, pp. 93–111, 1 2021.

[40] R. Pan, P. Natarajan, F. Baker, and G. White, "Proportional integral controller enhanced (pie): A lightweight control scheme to address the bufferbloat problem," Tech. Rep., 2017.

[41] R. Pan, P. Natarajan, C. Piglione, M. S. Prabhu, V. Subramanian, F. Baker, and B. VerSteeg, "Pie: A lightweight control scheme to address the bufferbloat problem," in *2013 IEEE 14th international conference on high performance switching and routing (HPSR)*. IEEE, 2013, pp. 148–155.

[42] K. Winstein and H. Balakrishnan, "Tcp ex machina: Computer-generated congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 123–134, 2013.

[43] M. Dong, Q. Li, D. Zarchy, P. B. Godfrey, and M. Schapira, "{PCC}: Re-architecting congestion control for consistent high performance," in *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, 2015, pp. 395–408.

[44] M. Dong, T. Meng, D. Zarchy, E. Arslan, Y. Gilad, B. Godfrey, and M. Schapira, "{PCC} vivace: Online-learning congestion control," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 343–356.

[45] T. Meng, N. R. Schiff, P. B. Godfrey, and M. Schapira, "Pcc proteus: Scavenger transport and beyond," in *Proceedings of the Annual conference of the ACM Special Interest Group on Data Communication on the applications, technologies, architectures, and protocols for computer communication*, 2020, pp. 615–631.

[46] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, pp. 1–12.

[47] V. Arun and H. Balakrishnan, "Copa: Practical delay-based congestion control for the internet," in *15th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 18)*, 2018, pp. 329–342.

[48] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 {USENIX} Annual Technical Conference ({USENIX}{ATC} 18)*, 2018, pp. 731–743.

[49] L. Jia, T. Huang, and L. Sun, "Zixia: A reinforcement learning approach via adjusted ranking reward for internet congestion control," in *ICC 2022-IEEE International Conference on Communications*. IEEE, 2022, pp. 365–370.

[50] H. Jiang, Q. Li, Y. Jiang, G. Shen, R. Sinnott, C. Tian, and M. Xu, "When machine learning meets congestion control: A survey and comparison," *Computer Networks*, vol. 192, p. 108033, 2021.

[51] J. Sun and M. Zukerman, "An adaptive neuron aqm for a stable internet," in *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet: 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007. Proceedings 6*. Springer, 2007, pp. 844–854.

[52] Q. Yan and Q. Lei, "A new active queue management algorithm based on self-adaptive fuzzy neural-network pid controller," in *2011 International Conference on Internet Technology and Applications*. IEEE, 2011, pp. 1–4.

[53] A. P. Silva, K. Obraczka, S. Burleigh, and C. M. Hirata, "Smart congestion control for delay-and disruption tolerant networks," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2016, pp. 1–9.

[54] S. Masoumzadeh, G. Taghizadeh, K. Meshgi, and S. Shiry, "Deep blue: A fuzzy q-learning enhanced active queue management scheme," in *2009 International Conference on Adaptive and Intelligent Systems*. IEEE, 2009, pp. 43–48.

[55] A. P. Silva, K. Obraczka, S. Burleigh, and C. M. Hirata, "Smart congestion control for delay-and disruption tolerant networks," in *2016 13th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*. IEEE, 2016, pp. 1–9.

[56] C. Zhou, D. Di, Q. Chen, and J. Guo, "An adaptive aqm algorithm based on neuron reinforcement learning," in *2009 IEEE International Conference on Control and Automation*. IEEE, 2009, pp. 1342–1346.

[57] M. Shahbaz, S. Choi, B. Pfaff, C. Kim, N. Feamster, N. McKeown, and J. Rexford, "Pisces: A programmable, protocol-independent software switch," in *Proceedings of the 2016 ACM SIGCOMM Conference*, ser. SIGCOMM '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 525–538. [Online]. Available: https://doi.org/10.1145/2934872.2934886

[58] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, p. 87–95, Jul. 2014. [Online]. Available: https://doi.org/10.1145/2656877.2656890

[59] J. Gomez, E. F. Kfoury, J. Crichigno, and G. Srivastava, "A survey on tcp enhancements using p4-programmable devices," *Computer Networks*, vol. 212, p. 109030, 2022.

[60] A. Feldmann, B. Chandrasekaran, S. Fathalli, and E. N. Weyulu, "P4-enabled network-assisted congestion feedback: A case for nacks," in *Proceedings of the 2019 Workshop on Buffer Sizing*, 2019, pp. 1–7.

[61] M. Handley, C. Raiciu, A. Agache, A. Voinescu, A. W. Moore, G. Antichi, and M. Wójcik, "Re-architecting datacenter networks and stacks for low latency and high performance," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 29–42.

[62] M. Kang, G. Yang, Y. Yoo, and C. Yoo, "Proactive congestion avoidance for distributed deep learning," *Sensors*, vol. 21, no. 1, p. 174, 2020.

[63] Shahzad, E.-S. Jung, J. Chung, and R. Kettimuthu, "Enhanced explicit congestion notification (eecn) in tcp with p4 programming," 1 2020. [Online]. Available: https://www.osti.gov/biblio/1669092

[64] A. Laraba, J. François, S. R. Chowdhury, I. Chrisment, and R. Boutaba, "Mitigating tcp protocol misuse with programmable data planes," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 760–774, 2021.

[65] R. Kundel, J. Blendin, T. Viernickel, B. Koldehofe, and R. Steinmetz, "P4-codel: Active queue management in programmable data planes," in *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2018, pp. 1–4.

[66] I. Kunze, M. Gunz, D. Saam, K. Wehrle, and J. Rüth, "Tofino+ p4: A strong compound for aqm on high-speed networks?" in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. IEEE, 2021, pp. 72–80.

[67] R. Kundel, A. Rizk, J. Blendin, B. Koldehofe, R. Hark, and R. Steinmetz, "P4-codel: Experiences on programmable data plane hardware," in *ICC 2021-IEEE International Conference on Communications*. IEEE, 2021, pp. 1–6.

[68] C. Papagianni and K. De Schepper, "Pi2 for p4: An active queue management scheme for programmable data planes," in *Proceedings of the 15th International Conference on Emerging Networking Experiments and Technologies*, 2019, pp. 84–86.

[69] L. Toresson, "Making a packet-value based aqm on a programmable switch for resource-sharing and low latency," 2021.

[70] N. K. Sharma, A. Kaufmann, T. E. Anderson, A. Krishnamurthy, J. Nelson, and S. Peter, "Evaluating the power of flexible packet processing for network resource allocation." in *NSDI*, 2017, pp. 67–82.

[71] A. Mushtaq, R. Mittal, J. McCauley, M. Alizadeh, S. Ratnasamy, and S. Shenker, "Datacenter congestion control: Identifying what is essential and making it practical," *ACM SIGCOMM Computer Communication Review*, vol. 49, no. 3, pp. 32–38, 2019.

[72] M. Menth, H. Mostafaei, D. Merling, and M. Häberle, "Implementation and evaluation of activity-based congestion management using p4 (p4-abc)," *Future Internet*, vol. 11, no. 7, p. 159, 2019.

[73] A. G. Alcoz, A. Dietmüller, and L. Vanbever, "Sp-pifo: Approximating push-in first-out behaviors using strict-priority queues." in *NSDI*, 2020, pp. 59–76.

[74] C. Cascone, N. Bonelli, L. Bianchi, A. Capone, and B. Sansò, "Towards approximate fair bandwidth sharing via dynamic priority queuing," in *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE, 2017, pp. 1–6.

[75] B. Turkovic, S. Biswal, A. Vijay, A. Hüfner, and F. Kuipers, "P4qos: Qos-based packet processing with p4," in *2021 IEEE 7th International Conference on Network Softwarization (NetSoft)*. IEEE, 2021, pp. 216–220.

[76] C. Chen, H.-C. Fang, and M. S. Iqbal, "Qostcp: Provide consistent rate guarantees to tcp flows in software defined networks," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[77] S. Sengupta, H. Kim, and J. Rexford, "Continuous in-network round-trip time monitoring," 2022, Conference paper, p. 473 – 485, cited by: 4; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85138021661&doi=10.1145%2f3544216.3544222&partnerID=40&md5=031277528a83fcee7fa5dc6865da9a1b

[78] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic router's buffer sizing using passive measurements and p4 programmable switches," 2021, Conference paper, cited by: 2. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85127230361&doi=10.1109%2fGLOBECOM46510.2021.9685160&partnerID=40&md5=0fcd35e0d1735dbd0fa50347867dc2bb

[79] S. Liu, X. Lin, Z. Guo, Y. Wang, M. A. Serhani, and Y. Xu, "Optimizing flow completion time via adaptive buffer management in data center networks," in *50th International Conference on Parallel Processing*, 2021, pp. 1–10.

[80] J. Zhang, W. Bai, and K. Chen, "Enabling ecn for datacenter networks with rtt variations," in *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*, 2019, pp. 233–245.

[81] C. Raiciu, S. Barre, C. Pluntke, A. Greenhalgh, D. Wischik, and M. Handley, "Improving datacenter performance and robustness with multipath tcp," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 266–277, 2011.

[82] M. Al-Saadi, A. Khan, V. Kelefouras, D. J. Walker, and B. Al-Saadi, "Unsupervised machine learning-based elephant and mice flow identification," in *Intelligent Computing: Proceedings of the 2021 Computing Conference, Volume 2*. Springer, 2021, pp. 357–370.

[83] J. Crichigno, E. Bou-Harb, and N. Ghani, "A Comprehensive Tutorial on Science DMZ," *IEEE Communications Surveys Tutorials*, vol. 21, no. 2, pp. 2041–2078, 2019.

[84] S. Liu, X. Lin, Z. Guo, Y. Wang, M. A. Serhani, and Y. Xu, "Optimizing flow completion time via adaptive buffer management in data center networks," in *Proceedings of the 50th International*

*Conference on Parallel Processing*, ser. ICPP '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://doi.org/10.1145/3472456.3472507

[85] H. Zhang, L. Lu, L. Xiao, and W. Dou, "Preferential bandwidth allocation for short flows with active queue management," in *Network and Parallel Computing: IFIP International Conference, NPC 2005, Beijing, China, November 30-December 3, 2005. Proceedings*. Springer, 2005, pp. 303–309.

[86] K. Avrachenkovt, U. Ayesta, P. Brown, and E. Nyberg, "Differentiation between short and long tcp flows: Predictability of the response time," in *IEEE INFOCOM 2004*, vol. 2. IEEE, 2004, pp. 762–773.

[87] D. M. Divakaran, E. Altman, and P. Vicat-Blanc Primet, "Size-based flow-scheduling using spike-detection," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6751 LNCS, p. 331 – 345, 2011, cited by: 8. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-79959365809&doi=10.1007%2f978-3-642-21713-5_24&partnerID=40&md5=3967c7d7912bf43fb04972ea5f5eb2dc

[88] F. Carpio, A. Engelmann, and A. Jukan, "Diffflow: Differentiating short and long flows for load balancing in data center networks," in *2016 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2016, pp. 1–6.

[89] P. Dong, W. Yang, W. Tang, J. Huang, H. Wang, Y. Pan, and J. Wang, "Reducing transport latency for short flows with multipath tcp," *Journal of Network and Computer Applications*, vol. 108, pp. 20–36, 2018.

[90] W. xi Liu, J. Cai, Q. C. Chen, and Y. Wang, "Drl-r: Deep reinforcement learning approach for intelligent routing in software-defined data-center networks," *Journal of Network and Computer Applications*, vol. 177, p. 102865, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804520303313

[91] W. Zang, Z. Jin, and J. Lan, "An sdn based fast rerouting mechanism for elephant flows in dcn," in *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, 2017, pp. 363–366.

[92] H. T. Zaw and A. H. Maw, "Elephant flow detection and delay-aware flow rerouting in software-defined network," vol. 2018-January, 2017, Conference paper, p. 1 – 6, cited by: 6. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85049477773&doi=10.1109%2fICITEED.2017.8250487&partnerID=40&md5=7138c9a7671fa4792cd37c8e6484751b

[93] X. Hu, X. Miao, and M. Yuan, "Research on elephant flow detection method based on information entropy and improved random forest," vol. 12176, 2022, Conference paper, cited by: 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85132810654&doi=10.1117%2f12.2636382&partnerID=40&md5=bd859f1069e92187d4f4032fed44ff12

[94] W.-X. Liu, J. Cai, Y. Wang, Q. C. Chen, and J.-Q. Zeng, "Fine-grained flow classification using deep learning for software defined data center networks," *Journal of Network and Computer Applications*, vol. 168, 2020, cited by: 11. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088921464&doi=10.1016%2fj.jnca.2020.102766&partnerID=40&md5=e4ce92f1934e8fb0abcbf04d27a5499c

[95] R. Durner and W. Kellerer, "Network function offloading through classification of elephant flows," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, p. 807 – 820, 2020, cited by: 10; All Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85081322489&doi=10.1109%2fTNSM.2020.2976838&partnerID=40&md5=c64fd43d4fb6c331703b8b243847f1c0

[96] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," in *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 281–292.

[97] C. Villamizar and C. Song, "High Performance TCP in ANSNET," *SIGCOMM Comput. Commun. Rev.*, vol. 24, no. 5, p. 45–60, Oct. 1994.

[98] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing Router Buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 4, p. 281–292, 2004.

[99] A. Vishwanath, V. Sivaraman, and M. Thottan, "Perspectives on Router Buffer Sizing: Recent Results and Open Problems," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 2, p. 34–39, Mar. 2009.

[100] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: congestion-based congestion control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.

[101] E. F. Kfoury, J. Gomez, J. Crichigno, and E. Bou-Harb, "An emulation-based evaluation of TCP BBRv2 Alpha for wired broadband," *Computer Communications*, vol. 161, pp. 212 – 224, 2020.

[102] M. Enachescu, Y. Ganjali, A. Goel, N. McKeown, and T. Roughgarden, "Part III: Routers with Very Small Buffers," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, p. 83–90, Jul. 2005.

[103] P. Raja and G. Raina, "Delay Based Feedback, Transport Protocols and Small Buffers," in *Proceedings of the Nineteenth International Workshop on Quality of Service*, ser. IWQoS '11. IEEE Press, 2011.

[104] S. Abbasloo, C.-Y. Yen, and H. J. Chao, "Classic Meets Modern: A Pragmatic Learning-Based Congestion Control for the Internet," in *Proc. Conf. ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '20. New York, NY, USA: ACM, 2020, p. 632–647.

[105] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS operating systems review*, vol. 42, no. 5, pp. 64–74, 2008.

[106] N. Beheshti, P. Lapukhov, and Y. Ganjali, "Buffer Sizing Experiments at Facebook," in *Proc. Workshop Buffer Sizing*, ser. BS '19. New York, NY, USA: ACM, 2019.

[107] B. Spang, B. Walsh, T.-Y. Huang, T. Rusnock, J. Lawrence, and N. McKeown, "Buffer Sizing and Video QoE Measurements at Netflix," in *Proceedings of the 2019 Workshop on Buffer Sizing*, ser. BS '19. New York, NY, USA: Association for Computing Machinery, 2019.

[108] X. Zhang, N. Gu, J. Su, and K. Ren, "DFTCP: A TCP-friendly delay-based high-speed TCP variant," in *2016 International Conference on Networking and Network Applications (NaNA)*, 2016, pp. 273–278.

[109] T. Zhu, X. Qin, L. Chen, X. Chen, and G. Wei, "wBBR: A Bottleneck Estimation-Based Congestion Control for Multipath TCP," in *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall)*, 2018, pp. 1–5.

[110] F. Ciaccia, I. Romero, O. Arcas-Abella, D. Montero, R. Serral-Gracià, and M. Nemirovsky, "SABES: Statistical Available Bandwidth EStimation from passive TCP measurements," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 743–748.

[111] S. K. Khangura and S. Akın, "Measurement-Based Online Available Bandwidth Estimation Employing Reinforcement Learning," in *2019 31st International Teletraffic Congress (ITC 31)*, 2019, pp. 95–103.

[112] J. Crichigno, N. Ghani, J. Khoury, W. Shu, and M.-Y. Wu, "Dynamic routing optimization in WDM networks," in *2010 IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[113] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[114] R. Hofstede, A. Pras, A. Sperotto, and G. D. Rodosek, "Flow-based compromise detection: Lessons learned," *IEEE security & privacy*, vol. 16, no. 1, pp. 82–89, 2018.

[115] M. Golling, R. Hofstede, and R. Koch, "Towards multi-layered intrusion detection in high-speed networks," in *2014 6th International Conference on Cyber Conflict (CyCon 2014)*. IEEE, 2014, pp. 191–206.

[116] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast ip networks," in *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE'05)*. IEEE, 2005, pp. 172–177.

[117] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An empirical evaluation of entropy-based traffic anomaly detection," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, 2008, pp. 151–156.

[118] F. Farina, P. Szegedi, and J. Sobieski, "Géant world testbed facility: federated and distributed testbeds as a service facility of géant," in *2014 26th International Teletraffic Congress (ITC)*. IEEE, 2014, pp. 1–6.

[119] P. Bereziński, B. Jasiul, and M. Szpyrka, "An entropy-based network anomaly detection method," *Entropy*, vol. 17, no. 4, pp. 2367–2408, 2015.

[120] C. Callegari, S. Giordano, and M. Pagano, "Entropy-based network anomaly detection," in *2017 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, 2017, pp. 334–340.

[121] I. Homem, P. Papapetrou, and S. Dosis, "Entropy-based prediction of network protocols in the forensic analysis of dns tunnels," *arXiv preprint arXiv:1709.06363*, 2017.

[122] Y. Gokcen, V. A. Foroushani, and A. N. Z. Heywood, "Can we identify nat behavior by analyzing traffic flows?" in *2014 IEEE Security and Privacy Workshops*. IEEE, 2014, pp. 132–139.

[123] J. Šuh, Ž. Bojović, M. Despotović-Zrakić, Z. Bogdanović, and A. Labus, "Designing a course and infrastructure for teaching software-defined networking," *Computer Applications in Engineering Education*, vol. 25, no. 4, pp. 554–567, 2017.

[124] C. Bouras, A. Kollia, and A. Papazois, "Teaching network security in mobile 5g using onos sdn controller," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, July 2017, pp. 465–470.

[125] M. Bima, O. Inalegwu, T. Folorunso, I. Oyefolahan, and S. Bala, "Software defined network (sdn) based virtual laboratory: An implementation." in *International Conference on Information Communication Technology and Its Applications (ICTA)*, 11 2016, pp. 232–235.

[126] N. Yalcin, Y. Altun, and U. Kose, "Educational material development model for teaching computer network and system management," *Computer Applications in Engineering Education*, vol. 23, no. 4, pp. 621–629.

[127] E. Salib and J. Lester, "Software defined network (sdn) based virtual laboratory: An implementation." in *2018 American Society for Engineering Education (ASEE) Annual Conference and Exposition*, 06 2018, pp. 232–235.

[128] J. Alcorn, S. Melton, and C. E. Chow, "Sdn on-the-go (otg) physical testbed," in *2017 IEEE Conference on Dependable and Secure Computing*, Aug 2017, pp. 202–208.

[129] S. M. Dascalu, Y. L. Varol, F. C. Harris, and B. T. Westphal, "Computer science capstone course senior projects: from project idea to prototype implementation," in *Proceedings Frontiers in Education 35th Annual Conference*, Oct 2005, pp. S3J–1.

[130] M. T. Hagan and H. B. Demuth, "Neural networks for control," in *Proc. ACC 1999*, vol. 3. IEEE, 1999, pp. 1642–1656.

[131] B. Kamanditya and B. Kusumoputro, "Elman recurrent neural networks based direct inverse control for quadrotor attitude and altitude control," in *2020 Int. Conf. Intelligent Eng. and Management*, 2020, pp. 39–43.

[132] B. Y. Suprapto and B. Kusumoputro, "A comparison of back propagation neural network and elman recurrent neural network algorithms on altitude control of heavy-lift hexacopter based on direct inverse control," in *2018 Int. Conf. ICECOS*, 2018, pp. 79–84.

[133] H. Alshareefi, C. Lupu, S. Olteanu, and L. Ismail, "Design and simulation of adaptive neuro-fuzzy inference system inverse controller for a coupled tank system," in *2021 10th International Conference on ENERGY and ENVIRONMENT (CIEM)*, 2021, pp. 1–5.

[134] T. Duriez, S. L. Brunton, and B. R. Noack, *Machine learning control-taming nonlinear dynamics and turbulence*. Springer, 2017.

[135] A. Jayachitra and R. Vinodha, "Genetic algorithm based pid controller tuning approach for continuous stirred tank reactor," *Advances in Artificial Intelligence*, vol. 2014, pp. 9–9, 2015.

[136] A. Mirzal, S. Yoshii, and M. Furukawa, "Pid parameters optimization by using genetic algorithm," *arXiv preprint arXiv:1204.0885*, 2012.

[137] J. Zhao and M. Xi, "Self-tuning of pid parameters based on adaptive genetic algorithm," in *IOP conference series: materials science and engineering*, vol. 782, no. 4. IOP Publishing, 2020, p. 042028.

[138] A. F. Gad, "Pygad: An intuitive genetic algorithm python library," 2021.

[139] S. Hemminger *et al.*, "Network emulation with netem," in *Linux conf au*, vol. 5. Citeseer, 2005, p. 2005.

[140] K. Chard, S. Tuecke, and I. Foster, "Globus: Recent enhancements and future plans," in *Proc. XSEDE16 Conf. Diversity, Big Data, and Science at Scale*, 2016, pp. 1–8.

[141] X. Chen, H. Kim, J. M. Aman, W. Chang, M. Lee, and J. Rexford, "Measuring TCP round-trip time in the data plane," in *Proc. Workshop Secure Programmable Net Infrastructure*, 2020, pp. 35–41.

[142] "P4_16 portable switch architecture (psa)," 2021. [Online]. Available: https://p4.org/p4-spec/docs/PSA.html

[143] C. Lee, C. Park, K. Jang, S. Moon, and D. Han, "Accurate latency-based congestion feedback for datacenters," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015, pp. 403–415.

[144] E. Kfoury, J. Crichigno, E. Bou-Harb, and G. Srivastava, "Dynamic Router's Buffer Sizing using Passive Measurements and P4 Programmable Switches," in *IEEE Global Comm. Conf. GLOBECOM*, 2021.

[145] F. Chollet *et al.* (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[146] M. A. et. al, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[147] A. Arfeen, K. Pawlikowski, D. McNickle, and A. Willig, "The role of the weibull distribution in modelling traffic in internet access and backbone core networks," *Journal of network and computer applications*, vol. 141, pp. 1–22, 2019.

[148] A. Varet and N. Larrieu, "Realistic network traffic profile generation: Theory and practice," *Computer and Information Science*, vol. 7, no. 2, pp. pp–1, 2014.

[149] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[150] B. Claise, "Rfc 3954: Cisco systems netflow services export version 9," 2004.

[151] P. Velan, J. Medková, T. Jirsík, and P. Čeleda, "Network traffic characterisation using flow-based statistics," in *NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium*, 2016, pp. 907–912.

[152] S. K. Routray and K. P. Sharmila, "Software defined networking for 5g," in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, Jan 2017, pp. 1–5.

[153] C. Xu, P. Li, and Y. Luo, "A programmable policy engine to facilitate time-efficient science dmz management," *Future Generation Computer Systems*, vol. 89, p. 515 – 524, 2018, cited by: 2. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85050259752&doi=10.1016%2fj.future.2018.07.016&partnerID=40&md5=c0d6abb2bbfff1ddd16a9319651f500e

[154] Y. Zhou, Y. Ren, X. Zhou, W. Yang, and Y. Qin, "A scientific data traffic scheduling algorithm based on software-defined networking," 2019, Conference paper, p. 62 – 67, cited by: 0. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85073537689&doi=10.1109%2fHPCC%2fSmartCity%2fDSS.2019.00024&partnerID=40&md5=ec8dfebdd540ff8a9f19ea8e883d7313

[155] V. Nagendra, V. Yegneswaran, P. Porras, and S. R. Das, "Coordinated dataflow protection for ultra-high bandwidth science networks," 2019, Conference paper, p. 568 – 583, cited by: 2; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85077814741&doi=10.1145%2f3359789.3359843&partnerID=40&md5=7e7e46e736529e58a16c12032d0b497c

[156] D. Nadig and B. Ramamurthy, "Securing large-scale data transfers in campus networks: Experiences, issues, and challenges," 2019, Conference paper, p. 29 – 32, cited by: 3; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85066125490&doi=10.1145%2f3309194.3309444&partnerID=40&md5=c1ea19716142b06ed9ecbdb85da6e09e

[157] D. N. Anantha and B. Ramamurthy, "Sciencesds: A novel software defined security framework for large-scale data-intensive science," 2017, Conference paper, p. 13 – 18, cited by: 4; All Open Access, Bronze Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85018272620&doi=10.1145%2f3040992.3040999&partnerID=40&md5=d1e19302c8bf3cec608468d6df1349da

[158] S. Miteff and S. Hazelhurst, "Nfshunt: A linux firewall with openflow-enabled hardware bypass," 2016, Conference paper, p. 100 – 106, cited by: 10. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-84963533515&doi=10.1109%2fNFV-SDN.2015.7387413&partnerID=40&md5=b1542e747b76bb4139e180a1350ebd4f

[159] S. Cosgrove, "Teaching software defined networking: It's not just coding," in *Teaching, Assessment, and Learning for Engineering (TALE), 2016 IEEE International Conference on*. IEEE, 2016, pp. 139–144.

[160] D. M. Batista, G. Blair, F. Kon, R. Boutaba, D. Hutchison, R. Jain, R. Ramjee, and C. E. Rothenberg, "Perspectives on software-defined networks: interviews with five leading scientists from the networking community," *Journal of Internet Services and Applications*, vol. 6, no. 1, p. 22, Oct 2015.

[161] M. Jou, C. K. Hung, and S.-H. Lai, "Application of challenge based learning approaches in robotics education," *International Journal of Technology and Engineering Education*, vol. 7, no. 2, pp. 17–18, 2010.

[162] D. Lártiga, N. Boettcher, Y. Prieto, and J. E. Pezoa, "Qosapp: Dynamic bandwith management for qos applications using openflow," in *2016 IEEE 37th Sarnoff Symposium*, Sep. 2016, pp. 1–2.

[163] Y. Prieto, J. E. Pezoa, N. Boettcher, and S. K. Sobarzo, "Increasing network reliability to correlated failures through optimal multiculture design," in *2017 CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, Oct 2017, pp. 1–6.

[164] Y. Prieto, C. Vega, J. E. Pezoa, and J. Crichigno, "Shared-risk-aware design for survivable migration in sdn environments," in *In Proc. IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV, USA*, Dec 2019, pp. 1–10.