Western Graduate&PostdoctoralStudies

Western University

Scholarship@Western

Electronic Thesis and Dissertation Repository

8-18-2021 9:45 AM

Leveraging Machine Learning Techniques towards Intelligent Networking Automation

Cesar A. Gomez, *The University of Western Ontario*

Supervisor: Shami, Abdallah, *The University of Western Ontario*
Co-Supervisor: Wang, Xianbin, *The University of Western Ontario*
A thesis submitted in partial fulfillment of the requirements for the Doctor of Philosophy degree in Electrical and Computer Engineering
© Cesar A. Gomez 2021

Follow this and additional works at: https://ir.lib.uwo.ca/etd

Part of the Artificial Intelligence and Robotics Commons, Data Science Commons, Digital Communications and Networking Commons, and the Systems and Communications Commons

Recommended Citation

Gomez, Cesar A., "Leveraging Machine Learning Techniques towards Intelligent Networking Automation" (2021). *Electronic Thesis and Dissertation Repository*. 8091.
https://ir.lib.uwo.ca/etd/8091

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Abstract

In this thesis, we address some of the challenges that the Intelligent Networking Automation (INA) paradigm poses. Our goal is to design schemes leveraging Machine Learning (ML) techniques to cope with situations that involve hard decision-making actions. The proposed solutions are data-driven and consist of an agent that operates at network elements such as routers, switches, or network servers. The data are gathered from realistic scenarios, either actual network deployments or emulated environments. To evaluate the enhancements that the designed schemes provide, we compare our solutions to non-intelligent ones. Additionally, we assess the trade-off between the obtained improvements and the computational costs of implementing the proposed mechanisms.

Accordingly, this thesis tackles the challenges that four specific research problems present. The first topic addresses the problem of balancing traffic in dense Internet of Things (IoT) network scenarios where the end devices and the Base Stations (BSs) form complex networks. By applying ML techniques to discover patterns in the association between the end devices and the BSs, the proposed scheme can balance the traffic load in a IoT network to increase the packet delivery ratio and reduce the energy cost of data delivery. The second research topic proposes an intelligent congestion control for internet connections at edge network elements. The design includes a congestion predictor based on an Artificial Neural Network (ANN) and an Active Queue Management (AQM) parameter tuner. Similarly, the third research topic includes an intelligent solution to the inter-domain congestion. Different from second topic, this problem considers the preservation of the private network data by means of Federated Learning (FL), since network elements of several organizations participate in the intelligent process. Finally, the fourth research topic refers to a framework to efficiently gathering network telemetry (NT) data. The proposed solution considers a traffic-aware approach so that the NT is intelligently collected and transmitted by the network elements.

All the proposed schemes are evaluated through use cases considering standardized networking mechanisms. Therefore, we envision that the solutions of these specific problems encompass a set of methods that can be utilized in real-world scenarios towards the realization of the INA paradigm.

Keywords

Autonomic Networking, Data-driven Networks, Intelligent Networking Automation, Intent-based Networking, Machine Learning, Zero-touch Network Management.

Summary for Lay Audience

Imagine living in a huge city where the traffic of the vehicles is solely controlled by officers: no traffic lights, no barricades, no separators, no signage, just traffic officers. What if a massive event is taking place near your home and you did not know? What if an accident occurs on a road you just merged onto? It is hard to visualize the flows of the vehicles going smoothly. Although the city had so many officers and their protocols were very well established, it would not be enough to regulate the vehicle flows properly. This research work is about something similar: the effective application of artificial intelligence methods to automate the control of Internet flows when the networks experience unforeseen situations. The proposed solutions allow the network administrators to manage some network tasks more efficiently, with minimal intervention, and focus on the situations where the human involvement is critical.

To my beloved wife Carolina and sweet daughters Sarita and Abril. I will never thank you enough for all your love, joyful support, and unselfish sacrifice. This endeavor would not have been possible without you.

To my loving parents Martha and Samuel. God bless you for all you have taught me to grow as a fulfilled person in this world. I hope to keep learning from you for many more years to come.

In memory of my exceptional grandparents Socorro, Gilma, Servio, and Samuel. Both your tenacity and tenderness run in my blood.

Acknowledgments

I would like to express my sincere appreciation to my supervisors, Dr. Abdallah Shami and Dr. Xianbin Wang, for all their guidance and support. I learned a lot from them and their expertise was crucial to the successful completion of my research journey during my doctoral studies. I feel so grateful and honored to have worked with them.

I would also like to thank the staff in the Department of Electrical and Computer Engineering at Western University, especially Stephanie Tigert, Courtney Harper, and Andrea Krasznai, for all their remarkable assistance and kindness to make the administrative tasks of my program easier to carry out. Correspondingly, many thanks to the team supporting the graduate matters in the Faculty of Engineering, including Dr. Kamran Siddiqui, Karen McDonald, and Whitney Barret. They were very open, approachable, and caring when I needed their help.

A special thank-you message to the staff of the School of Graduate and Postdoctoral Studies, particularly Dr. Linda Miller, Dr. Peter Simpson, and Dr. Mihaela Harmos, for their wonderful initiatives to support the PhD students at Western, such as the engaging Own Your Future program led by Dr. Julie Jonkhans. Many of their talks, workshops, and dialogues with other peers at their events served as a source of encouragement to tackle the inherent challenges of pursuing a PhD degree. In the same way, I feel grateful for the counselling services provided by the Career Education unit, which helped me explore the value and potential of my PhD degree both in and outside the academia. Special thanks to Jennifer Baytor and Dr. Steven Martin.

I am also thankful for all the support given by the members of the International and Exchange Student Centre, especially Sandra Pehilj and Fabiana Tepedino. They were always eager to help and their services were key to my successful adaptation as an international student at Western. Likewise, special thanks to Dr. Olga Kharytonava from the Western English Language Centre, a passionate language educator from whom I learned a lot to enhance my writing.

Additionally, I would like to thank my colleagues and research group coworkers for all their advice and willingness to help me, even when they were very busy doing their own research work. Special thanks to Dr. Fuad Shamieh, Dr. Khaled Alhazmi, Dr. Hassan Hawilo, Dr. Manar Jammal, Dr. Elena Uchiteleva, Dr. Abdallah Moubayed, Dr. Shree Sharma, Dr. Tianqi Yu, Golara Zafari, and Hessam Yousefi (RIP). Likewise, I am very thankful for the friendship offered by the colleagues I met at Western University, including Sabin Bhandari, Jose Masache, Sunanda Gamage, Ibrahim Shaer, Dimitrios Manias, Santiago Gomez, Mahsa Bataghva, Hasitha Wimalarathna, Sandra Lopez, Elham Okhovat, Sriyananda Mattaka, and Yuyan Zhao. Also, special mention to Dr. Julio Suarez for his unconditional amicability and mentorship on how to effectively navigate my doctorate pathway.

Last but not least, I would like to thank my family for their boundless love, trust, and patient support: Carolina, Abril, Sarita, Martha, Samuel, Ma. Laurita, Natalia, Daniela, Samuel F., Carlos, Amparo, Miguel, Johana, Ma. Jose, Silvana, Antonia, and Elkin. I thank God everyday for bringing you into my life.

Table of Contents

Abstract ii

Summary for Lay Audience iv

Acknowledgments vi

Table of Contents viii

List of Tables xi

List of Figures xii

List of Abbreviations xv

1. Introduction 1
   - 1.1. Motivation 1
   - 1.2. Preliminary Notions on INA 2
   - 1.3. Dissertation Contributions 8
   - 1.4. Dissertation Organization 9
2. Intelligent Load Balancing in IoT Networks 11
   - 2.1. Motivation 11
   - 2.2. Related Work 13
   - 2.3. A LoRaWAN Network Seen as a HetNet 15
   - 2.4. Proposed Scheme 17
     - 2.4.1. Data Preprocessing 17
     - 2.4.2. Pattern Analysis 19
     - 2.4.3. Classification Method for Association Biasing 21
     - 2.4.4. Decision-Making Model for Load Balancing 23
   - 2.5. Network Simulation Design 28
   - 2.6. Evaluation Results 30

2.6.1. PCA Patterns 31

2.6.2. Classifiers Outcomes 32

2.6.3. Network PDR Improvement 35

2.6.4. Network ECD Reduction 37

2.7. Summary 38

3. Intelligent Active Queue Management 39

3.1. Motivation 39

3.2. Intelligent AQM Design 41

3.2.1. Congestion Predictor 43

3.2.2. Q-learning based AQM Parameter Tuner 44

3.3. Evaluation Methodology and Results 46

3.3.1. Effects of Tuning AQM Parameters 46

3.3.2. Transferring the Predictor Model 48

3.3.3. Performance Evaluation of the Intelligent AQM 51

3.4. Summary 54

4. Federated Intelligence for Inter-Domain Congestion 55

4.1. Motivation 55

4.2. Related Work 60

4.3. Architecture of FIAQM 62

4.3.1. Federated Congestion Predictor 63

4.3.2. AQM Parameter Tuner 67

4.4. Experimentation Design 70

4.5. FIAQM performance evaluation 75

4.5.1. FCP algorithm predictions accuracy 75

4.5.2. Real-time AQM tuning with FIAQM 79

4.6. Summary 83

5. Efficient Network Telemetry based on Traffic Awareness 84
    5.1. Motivation 84
    5.2. Related Work 88
    5.3. TANT Traffic Classifier 89
        5.3.1. Inference Acceleration 97
    5.4. TANT Controller 100
    5.5. Experimentation and Evaluation Results 102
    5.6. Summary 106
6. Conclusion 108
    6.1. Dissertation Conclusions 108
    6.2. Discussion on the Findings and Limitations of this Thesis 109
    6.3. Future Work 112
7. References 114
8. Curriculum Vitae 132

List of Tables

Table 2.1. Simulation parameters for the evaluation of Packet Delivery Ratio and Energy Cost of Data Delivery. 30

Table 3.1. Buffer occupancy comparison 53

Table 4.1. Model parameters to be transferred for the FCP. 74

Table 4.2. Emulation parameters for the evaluation of the FIAQM scheme in real time 79

Table 4.3. Hyperparameters of the FIAQM'S learning modules. 82

Table 5.1. Time-related traffic features 91

Table 5.2. F1-Scores comparison after second feature selection 97

Table 5.3. Granularity Levels 102

List of Figures

Figure 1.1. Reference Model for an Autonomic Node. Adapted from [6]. 4

Figure 1.2. Mapping between architectural blocks and closed-loop automation stages in the ZSM framework. Adapted from [10]. 6

Figure 1.3. Architecture for ML in closed-loop automation. Adapted from [11]. 7

Figure 2.1. LoRaWAN network architecture. An end device may be associated with more than one gateway. Adapted from [40] 16

Figure 2.2. Locations of BS and their associated devices within the selected urban area. 18

Figure 2.3. Coverage approximation of BS based on data points, assuming isotropic radiation, and ideal propagation. 18

Figure 2.4. Example of a load-balancing decision to be made. 24

Figure 2.5. Algorithm for the traffic offloading decision. 27

Figure 2.6. Discovered patterns for BS after projecting the first two principal components. 31

Figure 2.7. Device-BS association comparison: (a) estimated coverage based on RSSI; (b) CRE based on biased association. 33

Figure 2.8. Classifiers performance comparison: (a) average classification accuracy; (b) average training time. 34

Figure 2.9. Network PDR improvement based on proposed scheme. 36

Figure 2.10. Computational time comparison for the MDPs. 36

Figure 2.11. Network EDC reduction based on proposed scheme. 37

Figure 3.1. Scenario for our stated problem. Edge routers aggregate end devices and connect to the core network through bottleneck links. 42

Figure 3.2. Effects of varying the target parameter in CoDel and FQ-CoDel algorithms on: a) Averaged mRTT. b) Averaged throughput. 47

Figure 3.3. Actual and predicted congestion obtained after: a) Pre-training over 100 epochs, using the CAIDA's dataset and time intervals of 100 ms. b) Re-training in one epoch, based on network emulation data in time intervals of 1 ms. 49

Figure 3.4. Cumulative power of the connection measured during the experiments in the emulation environment. The intelligent method is applied to CoDel and FQ-CoDel. All schemes utilize ECN 52

Figure 4.1. Example of an inter-domain communication scenario. 57

Figure 4.2. Typical scenario for the proposed FIAQM scheme. 60

Figure 4.3. The FIAQM architecture for inter-domain congestion control. Main modules are replicated within each border router. 62

Figure 4.4. LSTM network structure for the FIAQM's congestion predictor. 63

Figure 4.5. Implementation of the FIAQM for experimentation. 70

Figure 4.6. Preliminary tests for the Experimentation Design. a) Queues data at Border Router Domain 1. b) Effects of tuning FQ-CoDel target parameter on $\boldsymbol{m}\mathbf{RTT}$ and $\boldsymbol{Tput}$. 73

Figure 4.7. Actual congestion of the IXP queue and predicted congestion by the FCP. 76

Figure 4.8. Evaluation loss comparison between a centralized congestion predictor and the FCP algorithm. 77

Figure 4.9. Number of training rounds needed to reach the target loss by a centralized congestion predictor and the FCP algorithm. 78

Figure 4.10. Improvement over time provided by FIAQM in terms of congestion reduction and $\boldsymbol{Pc}$ growth. AQM tuning starts at 150 s. 81

Figure 5.1. TANT system overview. Each network element comprises a traffic classifier and an NT controller, which transmits the NT data to the NT engine........ 87

Figure 5.2. Correlation matrix of the selected features based on the Pearson coefficients. ... 94

Figure 5.3. Classifiers' scores comparison before and after first feature selection. 95

Figure 5.4. Feature ranking based on PFI calculation. 96

Figure 5.5. Computational resources used by the classifier algorithms for inferring in a single call. 99

Figure 5.6. Network scenario for the TANT use case evaluation........ 103

Figure 5.7. Evaluation resutls of TANT and its comparison against the classic INT. a) Network bandwidth overhead reduction per granularity level. b) Instantaneous and average network overhead measured during 1,200 seconds of network emulation. 105

List of Abbreviations

ACK	Acknowledged
AI	Artificial Intelligence
ANIMA	Autonomic Networking Integrated Model and Approach
ANN	Artificial Neural Network
AQM	Active Queue Management
AS	Autonomous System
BGP	Border Gateway Protocol
BS	Base Station
CAIDA	Center for Applied Internet Data Analysis
CE	Congestion Experienced
CIC	Canadian Institute for Cybersecurity
CoDel	Controlling Queue Delay
ConEx	Congestion Exposure
CP	Content Provider
CRE	Cell Range Expansion
CWR	Congestion Window Reduced
DDoS	Distributed Denial of Service
DE	Decision-making Element
DL	Downlink

DoS	Denial of Service
DT	Decision Trees
E2E	End-to-End
ECD	Energy Cost of Data Delivery
ECE	ECN-Echo
ECN	Explicit Congestion Notification
ET	Extra Trees
ETSI	European Telecommunications Standards Institute
FCP	Federated Congestion Predictor
FedAvg	Federated Averaging
FIAQM	Federated Intelligence for AQM
FL	Federated Learning
FN	False Negative
FP	False Positive
FQ-CoDel	Flow Queue - Controlling Queue Delay
GANA	Generic Autonomic Network Architecture
GNB	Gaussian Naive Bayes
HCL	Hierarchical Control Loop
HetNet	Heterogeneous Network
IAQM	Intelligent AQM

IAT	Inter-Arrival Time
ICMP	Internet Control Message Protocol
IETF	Internet Engineering Task Force
INA	Intelligent Network Automation
INT	In-Band Network Telemetry
INT-XD	INT-Export Data
IoT	Internet Of Things
IP	Internet Protocol
IPFIX	IP Flow Information Export
IRTF	Internet Research Task Force
ISG	Industry Specification Group
ISP	Internet Service Provider
ITU-T	International Telecommunication Union - Standardization Sector
IXP	Internet Exchange Point
LDA	Linear Discriminant Analysis
LightGBM	Light Gradient Boosting Machine
LoRaWAN	Long-Range Wide Area Network
LPWAN	Low-Power Wide Area Network
LR-SGD	Logistic Regression with Stochastic Gradient Descent Training
LSTM	Long Short-Term Memory

M2M	Machine-To-Machine
MAE	Mean Absolute Error
MBS	Macro Base Stations
MDP	Markov Decision Process
ME	Managed Entity
ML	Machine Learning
MLFO	Machine Learning Function Orchestrator
ML-ML	Machine Learning Meta Language
MLR	Multiple Logistic Regression
mRTT	Measured RTT
MSE	Mean Square Error
Multi-RAT	Multiple Radio Access Technology
NF	Network Function
NFV	Network Function Virtualization
NN	Neural Network
NT	Network Telemetry
ONNX	Open Neural Network Exchange
OODA	Observe, Orient, Decide, and Act
OPEX	Operational Expenditure
PBT	Postcard-Based Telemetry

PBT-M	Packet-Marking variation of PBT
PCA	Principal Components Analysis
PDP	Programmable Data Plane
PDR	Packet Delivery Ratio
PFI	Permutation Feature Importance
PIE	Proportional Integral Controller Enhanced
QDA	Quadratic Discriminant Analysis
QoS	Quality of Service
RAT	Radio Access Technology
RCA	Root Cause Analysis
RED	Random Early Detection
RF	Random Forest
RL	Reinforcement Learning
RMSE	Root Mean Squared Error
RNN	Recurrent Neural Network
RRUL	Real-Time Response Under Load
RSSI	Received Signal Strength Indicator
RTT	Round-Trip Time
SBS	Small Base Stations
SDN	Software Defined Networking

SFTP	Secure File Transfer Protocol
SINR	Signal-to-Interference-plus-Noise Ratio
SNMP	Simple Network Management Protocol
SNR	Signal-to-Noise Ratio
SSH	Secure Shell
SVM-SGD	Support Vector Machines with Stochastic Gradient Descent
TANT	Traffic-Aware Network Telemetry
TC	Traffic Control
TCP	Transmission Control Protocol
TFF	TensorFlow Federated
TN	True Negative
ToS	Type of Service
TP	True Positive
TTN	The Things Network
UDP	User Datagram Protocol
UL	Uplink
WAN	Wide Area Network
XGBoost	Extreme Gradient Boosting
ZSM	Zero-touch Network and Service Management

Chapter 1

Introduction

As a subfield of Artificial Intelligence (AI), Machine Learning (ML) is a discipline that aims to give a machine (or agent) the ability to execute tasks autonomously by detecting and extrapolating patterns, as well as adapting to new circumstances. Thus, an agent learns if it improves its performance after making observations (data samples) [1]. Researchers have applied ML techniques to solve a variety of non-trivial problems in many areas and the field of networking is not the exception. Because of the complexity and the dynamics of the networks, ML techniques can be successfully used to improve the performance of networking scenarios where optimal solutions are intractable to compute or difficult to represent through analytical models. Accordingly, the application of ML in networking includes a vast diversity of challenging tasks such as traffic prediction, traffic classification, routing, congestion control, resource management, load balancing, network scheduling, intrusion detection, and parameter adaptation, among others [2], [3].

1.1. Motivation

One of the most promising applications of ML in networking is to automate the networks in an intelligent way with minimal to no human intervention. As networks scale, they become more dynamic and complex systems. However, obtaining a closed-form function of these systems is non-trivial and analytical approximations to automate the networks may be imprecise. The closed-loop network automation refers to the notion of continuously evaluating real-time network conditions, traffic demands, and resource availability to determine the best placement of traffic for optimal service quality and resource utilization, according to the network operator policies. Consequently, the desired operation and performance improvement depend on the timely parameters’ adjustment and the changing network conditions.

Therefore, the notion of Intelligent Networking Automation (INA) has recently emerged as an answer to the challenge of managing large, complex, and very dynamic networks. The idea of having autonomous networks that can configure, monitor, and independently maintain themselves is not new. That is why the goals that INA pursues have been envisioned under different concepts, such as cognitive networking, autonomic networking, self-organized networks, knowledge-defined networks, intent-based networking, zero-touch networking, data-driven networking, and self-driving networks. INA will have a deep impact on the network management processes, as ML-based agents will carry out the tough networking tasks, allowing the operators to focus on the customers' needs and reduce their operational expenditure (OPEX). In addition, the combination of virtualized network infrastructure and online ML techniques will give the operators the flexibility to respond to real-time network parameters adjustment and scale their networks efficiently based on the changing business needs and the customers' demands.

Accordingly, the main objective of this chapter is to review the key components that the intelligent networks should have in terms of operations and management as well as presenting the concepts and frameworks that can make the INA a reality in the upcoming years. Also, we introduce how this dissertation contributes to the realization of that INA paradigm.

1.2. Preliminary Notions on INA

In order to achieve automation, the closed-loop control concept has been studied and applied to a variety of fields for decades, such as robotics and vehicle technologies. In networking, closed-loop control is used to automate tasks like resource allocation, performance optimization, devices management, fault analysis, etc. In a closed-loop control system, the controllers are connected in feedforward and feedback structures with physical elements and its components together determine the behavior of the overall system [4]. Thus, the closed-loop network automation refers to the notion of continuously evaluating real-time network conditions, traffic demands, and resource availability to determine the best placement of traffic for optimal service quality and resource utilization, according to the network operator policies [5].

The closed loops among network elements make the distinction between an automatic (or automated) network and an autonomic network. In the former, there are predefined processes that must be manually adjusted if the network environment changes. In the latter, the network processes act in a self-management fashion and can adapt to changing environments. Consequently, the RFC7575 from the Internet Research Task Force (IRTF) defines the concept of Autonomic Networking, which refers to the network capabilities of self-managing, i.e. self-configuring, self-protecting, self-healing, and self-optimizing [6]. An autonomic network consists of autonomic nodes, which exclusively employ autonomic functions: features that require no configuration and can adapt to a changing environment based on the information derived from self-knowledge, discovery, or intent. Thus, an autonomic node may have guidance by a central entity through intents, i.e. high-level policies used to operate the network.

The autonomic functions can be defined on a node level or on a system level. On a node level, the autonomic nodes interact each other to form feedback loops. On a system level, the central elements are also included in the feedback loops. These closed loops are a key aspect in autonomic networks and imply two-way negotiations between each pair or groups of peers involved in the loops. For this reason, a discovery phase is necessary before a closed-loop control can take place within an autonomic network.

The RFC7575 focuses on the intelligence of algorithms for node-level autonomic functions. This intelligence is realized by Autonomic Service Agents, which implement autonomic functions either entirely or partially (distributed functions). In this way, [6] presents the overview of a reference model for autonomic nodes, as depicted in Figure 1.1. Moreover, the Internet Engineering Task Force (IETF) has been working on an Internet-Draft that describes a reference model with more details, which is defined as Autonomic Networking Integrated Model and Approach (ANIMA) [7]. The ANIMA framework is an in-progress work and its architecture is well summarized in [8].

The Generic Autonomic Network Architecture (GANA) is another reference model, defined by the European Telecommunications Standards Institute (ETSI). In the GANA model, the ETSI defines a “blueprint model" with recommendations on the design and operational principles of autonomic decision-making elements (DEs), which are responsible for autonomic management and control of resources and parameters such as protocols, stacks, and mechanisms. Additionally, the DEs control the Managed Entities (MEs) in both physical and virtual network elements [9]. The GANA’s Decision Plane includes a Hierarchical Control Loop (HCL) architecture, in which DEs and MEs interact at different levels. In this way, the inferior DEs serve as the MEs of the superior DEs. Authors in [8] also review the GANA architecture although in less detail than the ANIMA model.

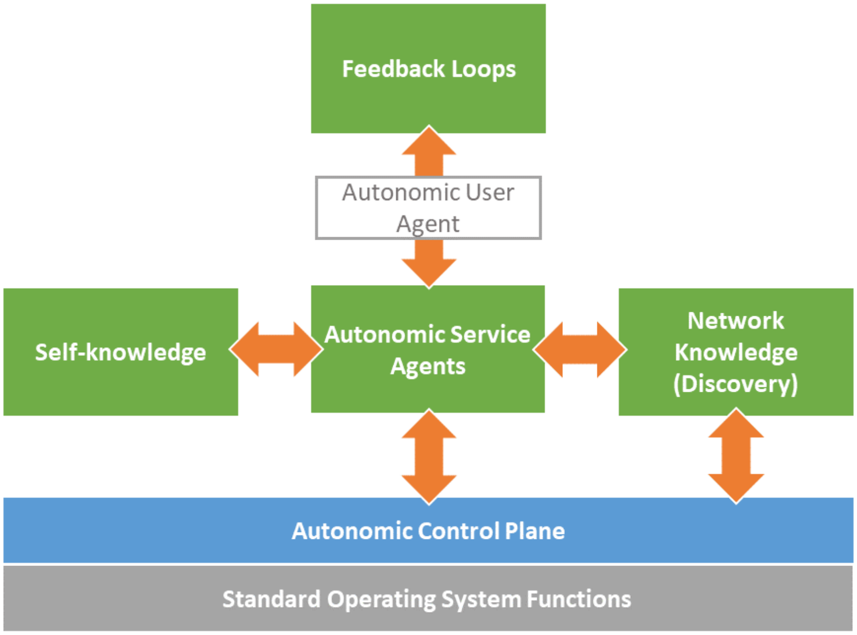


Figure 1.1. Reference Model for an Autonomic Node. Adapted from [6].

Another interesting initiative also from ETSI is the Zero-touch Network and Service Management (ZSM) Reference Architecture, which is a Group Specification by the Industry Specification Group (ISG) [10]. This specification presents an architecture for end-to-end (E2E) network automation, leveraging the principles of Network Functions Virtualization (NFV) [11], Software Defined Networking (SDN) [12], and cloud-native network services [13], as well as data-driven Artificial Intelligence algorithms [14], [15]. The ZSM architecture considers 13 principles to achieve its goal of full network automation, as follows: (1) modularity, (2) extensibility, (3) scalability, (4) model-driven and open interfaces, (5) closed-loop management automation, (6) support for stateless management functions, (7) resilience, (8) separation of concerns in management, (9) service composability, (10) intent-based interfaces, (11) functional abstraction, (12) simplicity, and (13) designed for automation.

We highlight the separation of concerns in management, as this architecture defines two domains: the Network Management Domain and the E2E Service Management Domain. The former manages resources and services delimited by technological or organizational boundaries and decouples the internal domain details from the outside world. The latter manages E2E services across multiple management domains and provides coordination between those domains. The internal domains of the network Management Domain includes: domain data collection, domain analytics, domain intelligence, domain orchestration, and domain control. Similarly, the E2E Service Management Domain comprises the E2E service data collection, E2E service data service analytics, E2E service intelligence, and E2E service orchestration. The management services in both domains can be provided and consumed by management functions, which are logical entities, deemed as either service consumers or service producers. In order to enable the interoperation and communication between management functions within and across management domains, the ZSM framework also outlines Domain Integration Fabric and Cross-domain Data services.

On the other hand, the principle of closed-loop management is the one that enables the E2E automation and zero-touch management of network services and infrastructures. Closing the management loop involves the transfer of information, knowledge, functions and operations such as analysis, learning, reasoning, planning, or decision-making capabilities. In order to achieve the closed-loop operation, ZSM considers a model that comprises the OODA stages: Observe, Orient, Decide, and Act. The management functions contribute with their respective management services capabilities at the respective OODA stages, as described in Figure 1.2.

Similarly, the International Telecommunication Union and its standardization sector (ITU-T) published a technical specification document that defines a unified architecture for ML in Fifth Generation and future networks [16]. This specification presents a set of requirements and constructs for the ML pipeline integration into evolving networks. This pipeline comprises the logical entities that can be combined to form analytics functions and each functionality in the pipeline is defined as a node. The possible nodes are: source of data (input for the ML function), collector of data, data pre-processor, ML model, policy (specific rules for network control), distributor (of ML outputs), and sink (target of the ML output, on which it takes action). The nodes are logical entities that are monitored and managed by a ML function orchestrator (MLFO) and hosted in a variety of network functions (NFs).

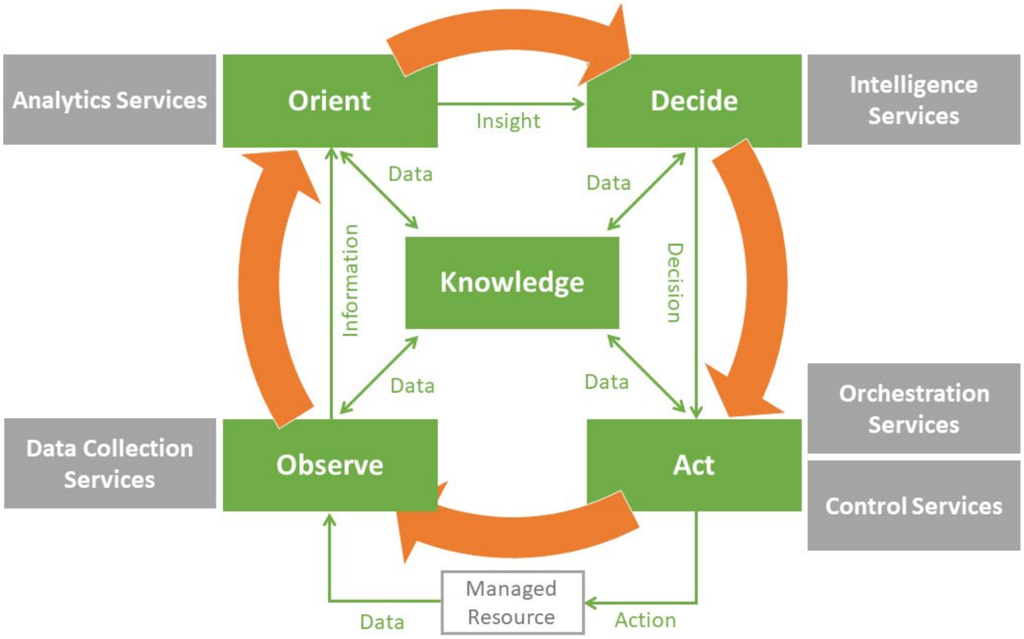


Figure 1.2. Mapping between architectural blocks and closed-loop automation stages in the ZSM framework. Adapted from [10].

The MLFO is a logical orchestrator that also selects and reselects the ML model based on its performance. Additionally, the MLFO is responsible for the placement of various ML pipeline nodes, based on the corresponding capabilities and constraints of the use cases, which are technology-independent and defined by intents. In other words, intents are mechanisms to specify the ML use case constructs and can employ ML meta language (ML-ML), which is needed to add the ML use case and the ML pipeline into the service design in a declarative manner.

The three main building blocks of the unified logical architecture comprises the management subsystem (which includes orchestration and various existing management entities), the multi-level ML pipeline (which uses the services of an MLFO for instantiation and setup), and the closed-loop subsystem (which allows the ML pipeline to adapt to dynamic network environments). Figure 1.3 depicts a simplified version of the proposed architecture to achieve closed-loop automation in operation and management on 5G networks. The management system is automated to promptly react to failures in the Network Function Virtualization (NFV). In this way, the network operator can promptly discover such failures, which result in gradually unstable behaviour before the process escalates into critical failure. Root Cause Analysis (RCA) is also important to properly convey the relationship information between failure type and location to the automation function. Consequently, the NFV Orchestrator is configured based on policy or workflows from the automation function. On the other hand, the ML pipeline is monitored and set up by the MLFO.

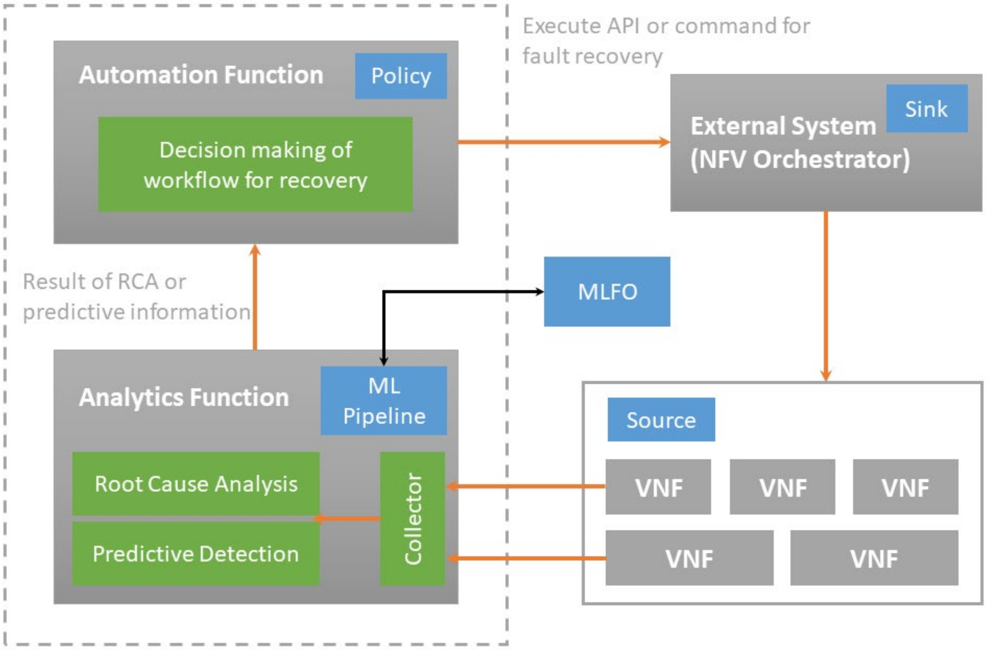

Execute API or command for fault recovery
Automation Function
Policy
Decision making of workflow for recovery
External System (NFV Orchestrator)
Sink
Result of RCA or predictive information
MLFO
Analytics Function
ML Pipeline
Root Cause Analysis
Predictive Detection
Collector
Source
VNF
VNF
VNF
VNF
VNF

Figure 1.3. Architecture for ML in closed-loop automation. Adapted from [11].

1.3. Dissertation Contributions

This dissertation presents the solution to several challenging networking situations that are aligned with the concepts on INA explained in the previous section. There are some commonalities among the proposed schemes, which leverage the application of ML methods to achieve different levels of INA. Those commonalties include the existence of a data collector and an autonomic agent that decides and takes actions based on the insights, knowledge discovery, or predictions derived from the collected data. Additionally, some designs also contemplate the orchestration of the intelligent mechanisms across different domains, meaning more than one single organization participating in the ML process.

On the other hand, we intend to introduce not only novel approaches to tackle the challenges that some networking scenarios pose, but also INA-oriented solutions that may eventually be implemented in real-world use cases. For this reason, all the frameworks introduced in this dissertation consider their application using standard protocols, specifications, or technologies. Furthermore, the data collection and knowledge discovery processes of the ML pipelines are performed in an online manner, so that the presented frameworks are evaluated through more realistic networking settings.

Accordingly, the main contributions of this thesis comprise the design of several INA solutions that aim at solving various networking problems, summarized as follows:

- Balancing the traffic in dense IoT networks, considering the Heterogeneous Network paradigm. To this end, we propose an ML scheme that learns from the available data of an operating IoT network to improve the network capacity in terms of the packet delivery ratio and the energy cost of data delivery.

- Proposing an ML-based scheme that address problem of congestion control for TCP/IP traffic, considering the AQM and ECN paradigms. The designed solution is fully compatible with existing TCP congestion control mechanisms and already deployed AQM techniques and improves the IP network capacity in terms of throughput and delay.

- A proof-of-concept study on non-static AQM. We demonstrate how the idea of dynamically tuning AQM parameters may boost the adoption of AQM mechanisms to mitigate the Internet's bufferbloat effect.

- An intelligent framework to control congestion over inter-domain links, which are not managed by a single party. The proposed solution is a multi-domain learning scheme in which local network data remains private. As in inter-domain scenarios privacy is a major concern, it allows the cooperation of two or more organizations to achieve common goals in terms of congestion by avoiding the share of raw data.

- The design of a flexible framework to achieve efficient Network Telemetry that can be adapted to a variety of telemetry schemes regardless their way of operation (in-band or out-of-band). The proposed mechanism can be intelligently adjusted to mitigate the network overhead that telemetry data collection and transmission produce.

- A set of methodological strategies to evaluate and implement solutions that employ ML algorithms making predictions based on real-world data and taking actions in real-time, such as the networking automation scenarios presented in this dissertation.

1.4. Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 describes the problem of load balancing in dense IoT networks. The chapter introduces some concepts on unsupervised and supervised ML as well as the applied techniques from those approaches to tackle the challenge of balancing traffic load. The utilized methodology is presented along with the results obtained from evaluating the presented solution through simulations. At the end of the chapter, the achieved improvements are discussed in terms of the packet delivery ratio and the energy cost of data delivery in a LoRaWAN network as a use case.

In Chapter 3, an intelligent scheme is proposed to address the problem of adjusting the parameters of standardized AQM schemes in dynamic TCP/IP networks. The chapter presents the application of a Deep Learning architecture to predict congestion on Internet links. Additionally, a RL method based on the Q-learning algorithm is utilized to adaptively change the AQM parameters of the links. The solution is evaluated through network emulations to consider a more realistic networking scenario and make the results reproducible in real networks. The end of the chapter shows that the intelligent method can enhance the TCP/IP connections in terms of latency and throughput.

In the same way, Chapter 4 presents a scenario where the intelligent AQM control needs to be achieved on links that interconnect two or more networks belonging to different organizations. To this end, the Federated Learning approach is applied, so that the network elements of each organization do not share private network data. The assessment of the proposed method presented at the end of the chapter shows that our proposed scheme is capable of adaptively changing the AQM parameters to reduce congestion on links that are shared by different domains, while preserving the privacy of each organization’s data.

Correspondingly, Chapter 5 introduces a novel method to collect and transmit network telemetry data by considering the types of traffic that a network element forwards. By means of supervised learning techniques, the proposed scheme determines the granularity of the telemetry data based on the classification of the flows that are being forwarded. Through network emulations, the solution is assessed and its results are discussed at the end of the chapter.

In Chapter 6, the major findings and limitations of this thesis are discussed. In addition, possible directions for future research work on the topics covered in this dissertation are explained. Finally, the references used in the research work of this thesis are presented and a brief Curriculum Vitae of the author is provided, including his publications to date.

Chapter 2

Intelligent Load Balancing in IoT Networks

2.1. Motivation

Thanks to the proliferation of Internet-connected wireless devices, the Internet of things (IoT) [17], [18] and the machine-to-machine (M2M) communications paradigms [19], highly dense cellular networks have emerged as a connectivity solution for large scale IoT applications. These wireless devices are diverse and comprise not only increasingly powerful devices like smart phones, but also tiny ones such as sensors, actuators, wearable electronics, etc. To alleviate the congestion in dense wireless networks, a number of solutions have been proposed. For instance, the idea of heterogeneous networks (HetNets) has been conceived. In a HetNet, the network infrastructure is supported by heterogeneous elements consisting of macro base stations (MBS), which provide a wide area coverage, and small base stations (SBS), that are meant to cover high traffic hotspots. The design of a cellular HetNet is based on a multi-tier topology, which features overlapped coverage between a tier of MBS and several sub-tiers of SBS. This design enhances the network capacity but at the cost of a challenging co-existence governing the network topology [20]. In fact, in urban areas, more SBS are added each year to the existing networks, creating a HetNet scenario where a wireless device may communicate with multiple BS, either MBS or SBS [21].

One of the most challenging design issues in HetNets is to achieve an optimal load balance among the base stations (BS), since the network traffic might be unevenly distributed. To this end, the association between devices and serving BS is a critical consideration. In homogeneous wireless networks, like the traditional cellular networks, a device is associated with the BS providing the strongest signal and, therefore, the association mechanisms are based on metrics such as signal-to-noise ratio (SNR) or received signal strength indicator (RSSI). However, this association method is not efficient for HetNets in terms of network capacity, since other critical aspects should be considered, such as, for example, the traffic load on the BS to be associated [22]. Device association methods based on signal metrics may lead to a major load imbalance in HetNets because MBS usually offer higher transmit power to devices than SBS. Consequently, load balancing methods for HetNets have been proposed by considering performance metrics like outage/coverage probability, spectrum efficiency, energy efficiency, uplink-downlink asymmetry, backhaul bottleneck, and mobility support [23]. Nevertheless, the achievement of a balanced HetNet is not easy and intelligent mechanisms that consider the traffic load and all related network conditions of BS are desirable due to the overall complexity of the process [24]. For this reason, artificial intelligence theory has been applied to overcome these kinds of challenges in complex systems like HetNets.

Load balancing in a HetNet may be performed by using either a single radio access technology (RAT) or multiple RAT (Multi-RAT). Multi-RAT techniques are aimed at taking advantage of load balancing between spectrum licensed technologies, e.g., cellular networks, and unlicensed ones, e.g., WiFi. However, the RAT selection algorithms, as well as the offloading mechanisms across cellular BS and WiFi access points, comprise an ambitious goal in terms of coordination and quality of service (QoS) [25]. In this work, we focus on the load balancing problem by considering a single RAT and its application to an actual IoT network. Specifically, the RAT used in this study is the LoRaWAN (long-range wide-area network) standard. In other words, we assume that the problem is delimited to the load balancing in an IoT network using a specific RAT. How to balance load considering more than one RAT is beyond of the scope of this work and it could be a promising research future work.

LoRaWAN is one of the most notable LPWAN (low-power wide area-network) technologies, alternative standards to conventional cellular networks, which have noteworthy expansions through IoT services providers [26]. As with other LPWAN technologies, LoRaWAN devices operate at a very low power, with long coverage (end devices can connect to a BS at a several-kilometers distance), and through a star topology, such as cellular networks [27], [28]. Another important characteristic is that LoRaWAN works in the unlicensed sub-GHz band, which is suitable for IoT applications in complex environments. However, the LoRaWAN protocol poses relevant challenges for dense networks regarding scalability and capacity. For example, in the default and most used class operation (Class A), LoRaWAN devices employ an uncoordinated access scheme (ALOHA) which might produce a collision avalanche in a large-scale network [29]. Therefore, optimization techniques are needed to allow reliable services and to avoid capacity drain in LoRaWAN networks with high densities of devices, such as those deployed in urban scenarios for smart cities.

In this work, we first show that an urban LoRaWAN network may be deemed as a HetNet. Hence, we address the problem of load balancing in a HetNet through appropriate machine learning (ML) techniques and we apply the proposed solution to improve the performance of a LoRaWAN network in a city. We further evaluate the performance of our solution in terms of the packet delivery ratio (PDR) and energy cost of data delivery (ECD) when the network has from a few to several thousands of end devices connected to it. Moreover, we expand our analysis to the case when devices request downlink traffic and not only the basic IoT scenario where uplink traffic is analyzed. The evaluation of our scheme is based on data collected from an actual network and its results illustrate that both PDR and energy cost are enhanced.

In the next sections, we review relevant works related to load balancing methods in HetNets (Section 2.2), we explain the factors to consider for an urban LoRaWAN network as a HetNet (Section 2.3), we describe our proposed scheme and its methods (Section 2.4), we give details about our network simulation design (Section 2.5), and we finally present the evaluation results (Section 2.6).

2.2. Related Work

A variety of approaches exists in the literature regarding the single RAT load balancing in HetNets. One of the most studied techniques is the cell range expansion (CRE): a mechanism to virtually expand an SBS range by adding a bias value to the power that a device receives from that SBS. In this way, instead of increasing the actual transmit power of an SBS, a virtual range expansion is performed so that a device will not connect to an MBS, but an SBS. However, to find an optimal bias value for minimizing the devices’ outage is a non-trivial problem and depends on several factors.

Accordingly, in [30] a scheme is proposed for the bias value optimization based on the Q-learning algorithm. The authors show that their method can decrease the number of outage devices and improve average throughput compared to non-learning schemes with a common bias value. Conversely, in [31] Ye et al. present a load-aware association method applied to CRE by considering two types of biasing factors, signal-to-interference-plus-noise ratio (SINR) and rate. The authors point out that the optimal biasing factors are nearly independent of the BS densities across tiers, but highly dependent on the per-tier transmit powers. Authors in [32] develop a clustering algorithm to classify BS into groups and present a central-aided distributed algorithm for adjusting the CRE bias. Their objective is to obtain a solution for the rate-related utility optimization problem based on local information. Thus, a central MBS is used to collect the information from the SBS, which determine their own CRE bias based on the shared central information. Similarly, authors in [33], [34] propose clustering techniques for optimizing the load balancing problem in HetNets.

Taking into account the energy efficiency, Ref. [35], [36] present techniques that are basically based on active/sleep schemes for multitier HetNets. In a similar manner, Muhammad et al. propose in [37] an association method that selectively mutes certain SBS. Then, end devices are covered by CRE for achieving load balancing in non-uniform HetNets, i.e., networks with SBS randomly deployed close to the edges of the MBS coverage, where the signals are weak. Contrary to the uniform case, their results show that biasing has distinct effects on the coverage and rate performance of a non-uniform HetNet. Lastly, authors in [38] propose a load balancing solution for a two-tier HetNet based on stochastic geometry. Their algorithm performs a CRE biasing to achieve an optimal SBS density regarding network energy efficiency.

Overall, biasing methods such as CRE are aimed at finding the appropriate bias values and at determining whether a specific BS should be considered or not for communication with a particular wireless device. An optimal decision of this association yields a network with balanced BS. This enhances the network performance in terms of capacity and energy efficiency, for instance, especially in scenarios with a large number of devices.

Although several ML algorithms have been presented in the literature to address the load balancing problem, they mainly focus on reinforcement learning techniques. Our method uses an unsupervised technique to discover the hidden pattern behind the selected features and a supervised technique to take advantage of the historical labeled data. Then, a supervised classifier is applied in order to accomplish a biasing scheme by contemplating metrics that are not directly related to signals strength. In this way, our model learns from data to predict a device-BS association without considering signal-based measurements. Additionally, our method employs a Markov Decision Process (MDP) to determine whether a BS needs to be balanced or not. For both techniques, the data are obtained from a real IoT LoRaWAN network deployed in an urban area, which is the use case scenario for our solution. To the best of our knowledge, this work is the first one that presents a solution to the load balancing problem applied to a LoRaWAN network.

2.3. A LoRaWAN Network Seen as a HetNet

As we have explained, the BS in a HetNet are dissimilar in terms of coverage and, therefore, BS are either MBS or SBS. We have also mentioned that LoRaWAN networks are cellular-like and are deployed following a star topology. However, unlike traditional cellular networks, LoRaWAN is an open standard and operates in the unlicensed bands, which allows rapid implementation of public and private networks. Then, in a smart city scenario where the priority of an IoT network might be capacity rather than communication range, the LoRaWAN access points are prone to being deployed in a non-homogeneous manner.

Moreover, it is also important to highlight that the LoRaWAN standard lets an end device be concurrently associated with more than one BS (i.e., gateway) [39], as shown in

Figure 2.1. We take into account this characteristic to evaluate the performance of our load balancing scheme. In this way, to be consistent with the standard, our goal is to determine what BS should transmit the downlink (DL) message to an end device, once an uplink (UL) message is received through more than one BS. This procedure is not defined by the LoRaWAN specifications and a network operator has to choose an optimal mechanism for it. Therefore, we consider a number of Class A end devices transmitting confirmed UL packets, i.e., packets that need to be acknowledged (ACK), and a network server that must make decisions on which gateways should relay the DL packets to end devices.

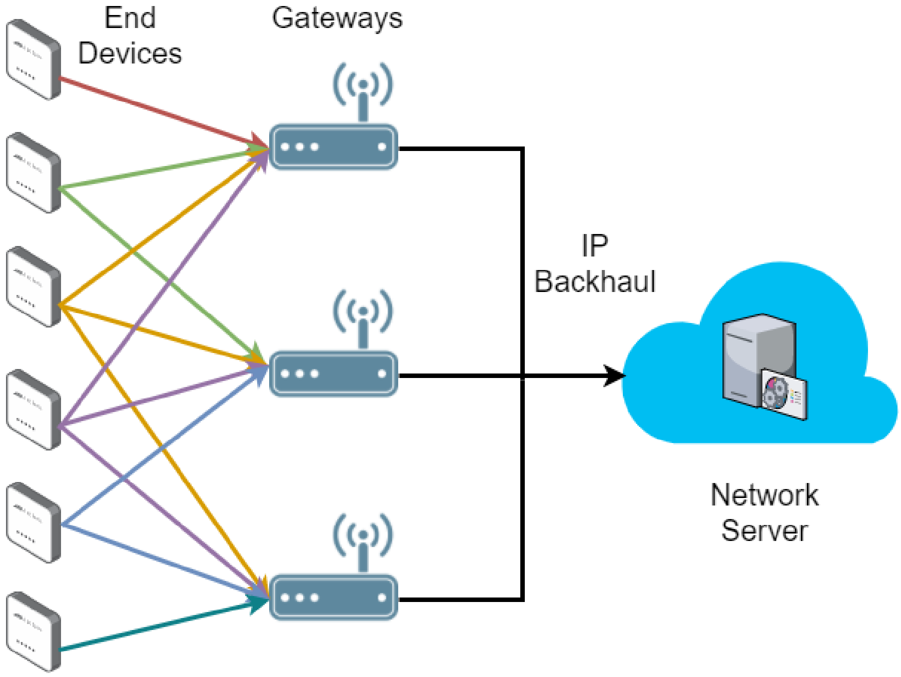


Figure 2.1. LoRaWAN network architecture. An end device may be associated with more than one gateway. Adapted from [40].

We also point out that our use case is based on data from The Things Network (TTN), a global collaborative LoRaWAN network crowdsourced by enthusiasts and with more than 4000 gateways [41]. Because of the nature of this IoT network, many gateways are randomly deployed, particularly in urban areas. Furthermore, the community members are encouraged to build their own gateways and private deployments might use a variety of available options in the marketplace, from macro gateways to pico gateways, e.g., [42]. As a result, the coverage areas of gateways are heterogeneous and overlap each other with diverse signal strength values. For these reasons, LoRaWAN networks such as TTN may be deemed as HetNets.

2.4. Proposed Scheme

Since our proposed solution for the load balancing problem is an ML-aided scheme, our methodology is data-driven and divided into four main stages: data preprocessing, pattern analysis, classification method, and decision-making model. The following subsections provide the details about each phase.

2.4.1. Data Preprocessing

In this stage we gather the historical data from an actual operating network. As mentioned in Section 2.3, the use case for our method is an IoT LoRaWAN network and that is why we take advantage of the TTN initiative. Specifically, we use the data available at the TTN Mapper website [43]. The TTN Mapper is an application fed by users with mobile devices and its main objective is to map the TTN gateways coverage by sending UL packets. For this work, we use the data dumped into tab-delimited files, which contain several fields that describe the connectivity status of the end devices at a given time and location, such as: node ID, timestamp, node address, address of the gateway the device is connected to, modulation in use, transmission data rate, SNR, RSSI, frequency, latitude, longitude, and altitude (the latter not available for all samples).

Since the files contain raw data, the first step is to clean and select the entries that are useful for our problem. To this end, we searched for data corresponding to an urban area taking into account the following considerations: (1) the BS with the highest number of received packets is the reference BS; (2) other BS are selected within a 10 km radius of the reference BS; (3) as end devices are mobile, only entries with location information of devices are considered; and (4) every BS is associated with two or more devices, thereby avoiding "dedicated" BS in the analysis. The resulting data is a subset of 261,576 samples, corresponding to seven BS. Figure 2.2 depicts the locations of the found BS and their devices in order to visualize how they are distributed and associated. Similarly, Figure 2.3 shows an idealization of the BS coverage based on their associated devices' locations. As can be seen, the data points show an urban scenario where some gateways behave like SBS and others like MBS. For example, BS 1 and BS 3 have shorter coverage ranges compared with the other gateways and their devices might be associated with BS 0 or BS 2, as well. Therefore, the selected data is suitable for our scheme and is consistent with our hypothesis of treating an urban dense IoT network as a HetNet.

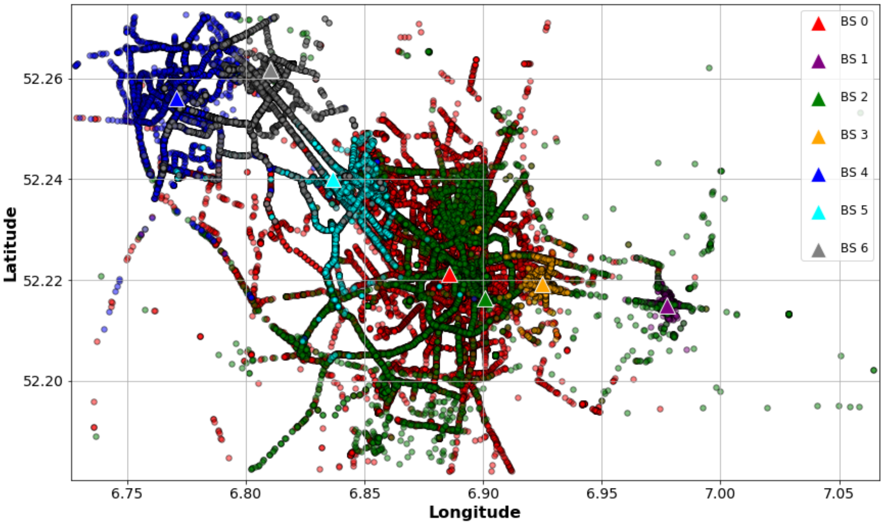


Figure 2.2. Locations of BS and their associated devices within the selected urban area.

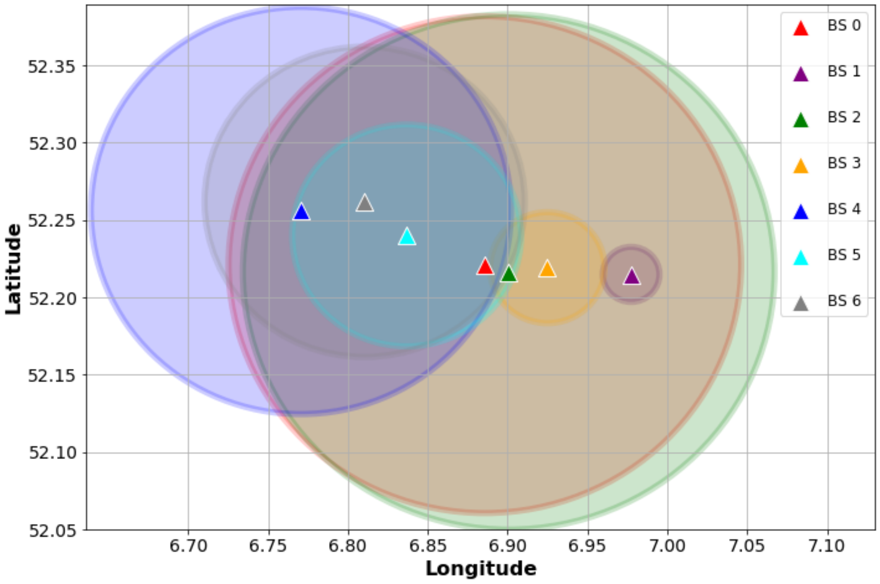


Figure 2.3. Coverage approximation of BS based on data points, assuming isotropic radiation, and ideal propagation.

Secondly, as our goal is to bias the device association to accomplish a load balancing, we extract several variables from data and waive the SNR and RSSI metrics. The main reason of doing so is to learn from the correlation among device's variables that are not directly influenced by the signal strength values. Thus, the features to be analyzed are some already available in the dataset such as frequency, data rate, latitude, and longitude, and others extracted from the timestamp field like time of the day, and day of the week. The idea of using these variables is to learn from their values as they describe the particular situation of a device at the moment that is successfully transmitting a packet to the BS.

2.4.2. Pattern Analysis

The purpose of this phase is to find out whether the extracted features provide differentiated patterns for each BS. To this end, we analyze the samples of the seven BS by using the principal components analysis (PCA). PCA is an unsupervised ML technique widely used for data visualization and feature selection. In this work, the main purpose to use PCA is to reduce the feature space to only two dimensions, so that the data of the end devices considering the selected features can be visualized.

PCA is a linear transform that maps the data into a lower dimensional space, known as the principal subspace, preserving as much data variance as possible, i.e., with minimum loss of information [44]. Since our features are frequency, data rate, latitude, longitude, time of the day, and day of the week, the original dimension of our data is a matrix $N \times D$, where $D = 6$ and $N = 261{,}576$. In this way, our objective is to project the data of each BS into two dimensions, i.e., $D = 2$, in order to visualize and verify that the extracted features do show a distinctive pattern. Therefore, for each BS there are N_k samples and PCA will produce two vectors of N_k elements, corresponding to the first two principal components. These vectors are computed as follows:

$$\mathbf{p}_c = \mathbf{w}_c^{\mathrm{T}} \mathbf{x}_k \tag{2.1}$$

where $c = \{1, 2\}$, $\mathbf{w}_c$ are the projection vectors, and $\mathbf{x}_k$ are the data subsets of each BS, i.e., $k = \{0, 1, 2, 3, 4, 5, 6\}$. In this case, the learning task is to choose $\mathbf{w}_c$ so that vectors $\mathbf{p}_c$ have the maximum variance. Thus, PCA determines vectors $\mathbf{w}_c$ by maximizing the variance in the projected space and by making them orthogonal, which means that $\mathbf{w}_1^{\mathrm{T}}\mathbf{w}_2 = 0$. This maximization problem can be solved through the incorporation of Lagrangian terms [44], that yields:

$$\mathbf{S}\mathbf{w}_c = \lambda_c \mathbf{w}_c \tag{2.2}$$

The pairs λ_c and $\mathbf{w}_c$ are the eigenvalues and the eigenvectors, respectively, of the covariance matrix $\mathbf{S}$, which is defined by(2.3):

$$\mathbf{S} = \frac{1}{N_k} \sum_{n=1}^{N_k} (\mathbf{x}_{k_n} - \bar{x}_k)\,(\mathbf{x}_{k_n} - \bar{x}_k)^{\mathrm{T}} \tag{2.3}$$

where $\bar{x}_k$ is the mean of sample subset $\mathbf{x}_k$.

Consequently, the variance will be maximum when $\mathbf{w}_1$ is equal to the eigenvector with the highest eigenvalue λ_1, giving as result the first principal component. The second principal component is given by selecting a new direction, so that $\mathbf{w}_2$ is orthogonal to $\mathbf{w}_1$ and equal to the eigenvector with the second highest eigenvalue λ_2.

Finally, it is also important to highlight that before performing the PCA, each feature is normalized by using the min-max scaling method (2.4):

$$z = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{2.4}$$

where z represents the normalized data points and x the original ones. The main objective of the scaling procedure is to have the values of all features within a range that is not too large, so that the variance maximization is not affected by their actual values [45]. Also, we have delimited $N_k = 10{,}000$ in order to have an equal number of samples for each BS and make a fairer comparison among their patterns.

2.4.3. Classification Method for Association Biasing

In this stage, we use the data to train the system and determine a biased association between a device and a particular BS. In our use case, we denote the device-BS association as the selection of a BS to relay DL packets. We do this distinction as a LoRaWAN device may be connected to several gateways to send UL packets to the Network Server, so biasing in UL makes no sense and is not consistent with the standard. On the other hand, we assume that the default DL association in TTN is based on signal strength, as suggested in [46]. Then, our purpose is to bias that DL path configuration, recognizing that a bidirectional traffic in a LoRaWAN network represents a more realistic scenario [47].

To bias the device-BS association, we take advantage of the labeled data by applying a supervised learning technique. Specifically, this technique is intended to perform a multi-class classification, since our goal is to predict the BS that should forward DL messages to an end device by avoiding the SNR and RSSI metrics. Hence, in our use case scenario we have seven classes, one per BS. In addition, the inputs of the classifier are the features contemplated for PCA and the labels, which are categorical values corresponding to one of the seven classes.

ML classification algorithms can be categorized into two types: probabilistic and non-probabilistic classifiers. The main difference between them is that non-probabilistic classifiers define a decision boundary to determine whether a prediction belongs or not to a specific class [48]. It means that a non-probabilistic classifier performs a hard classification: given the inputs values, the model yields only one class. On the other hand, a probabilistic classifier provides the probabilities of belonging to each class, instead of giving only one class as a result. Then, a probabilistic classifier produces a soft classification and does not define decision boundaries. As we want to bias the default device-BS association, it is desirable to find the probabilities of receiving DL packets through other BS. For this reason, we choose to use a probabilistic classifier. Additionally, these kinds of classifiers allow us to find the classification posterior probabilities, which can be used for our decision-making problem of load balancing.

In general, probabilistic classifiers are based on the Bayes' theorem to find the posterior class probabilities and determine the class membership for each new input $\mathbf{x}$ [44]. Thus, the posterior probabilities $p(C_k|\mathbf{x})$ are given by (2.5):

$$p(C_k|\mathbf{x}) = \frac{p(\mathbf{x}|C_k)p(C_k)}{p(\mathbf{x})} \tag{2.5}$$

where $p(\mathbf{x}|C_k)$ represents the class-conditional densities individually inferred for each class C_k, $p(C_k)$ are the prior class probabilities, which can be estimated from portions of the training subset, and $p(\mathbf{x})$ is found as follows (2.6):

$$p(\mathbf{x}) = \sum_k p\,(\mathbf{x}|C_k)p(C_k) \tag{2.6}$$

To select a specific classification method, we compare the accuracy and the computational time of several algorithms, such as multiple logistic regression (MLR), Gaussian naive Bayes (GNB), Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Decision Trees (DT). In addition, we include in our comparison some ensemble methods such as the Random Forests (RF), Extra Trees (ET) and a voting classifier. Details about the algorithms behind these classifiers can be found in [45], [48], [49].

Similar to the pattern analysis, an equal number of samples $N_k = 10{,}000$ are extracted for each class in order to have a balanced dataset and prevent the classifiers from being biased during the training process. To train and test the classifiers, the dataset is divided into two subsets: 80% and 20%, respectively. Based on these subsets, we also calculate the average accuracy of each classifier. In this way, we determine the true positive (TP), true negative (TN), false positive (FP), and false negative (FN) classification outcomes per class by comparing the predicted labels to the actual labels from the test subset samples. Note that a hard classification is needed for this comparison, therefore, we consider the class with the highest probability as the predicted label. Accordingly, the terms TP, TN, FP, and FN are derived from the confusion matrix, which summarizes the comparison: columns describe the outputs of predicted labels and rows, the actual labels. Thus, the value of TP for class 1, for example, is the number of predictions with label 1 that match the actual label 1, and the number of predictions that do not match is the value of FP. Similarly, TN is the number of predictions with other label different from label 1 that match any other actual label, and FN represents the otherwise case. Subsequently, the overall classifier accuracy with K classes can be calculated by macro-averaging the accuracy of the classes [50], i.e., all classes equally treated, as follows (2.7):

$$\text{Accuracy} = \frac{1}{K}\sum_{k=1}^{K} \frac{\text{TP}_k + \text{TN}_k}{\text{TP}_k + \text{TN}_k + \text{FP}_k + \text{FN}_k} \tag{2.7}$$

Additionally, we point out that before training the classifiers, the features are standardized by using the z-score Formula (2.8):

$$z = \frac{x - \mu}{\sigma} \tag{2.8}$$

where z is the standardized data point value, x is the original value, μ and σ are the mean and the standard deviation of each variable, respectively. In this way, the classifiers perform better with standard normally distributed data, i.e., with zero mean and unit variance [45].

Finally, we define $\mathbf{r}_k$ as the vector with the found probabilities after making a prediction for the biased association. Therefore, the values of $\mathbf{r}_k$ correspond to a device's probabilities of being associated with specific BS by waiving the signal-based features, and then $\sum_k r_k = 1$.

2.4.4. Decision-Making Model for Load Balancing

Our goal with the decision-making model is to achieve a load balance and, consequently, improve the network capabilities in terms of PDR and energy cost of data delivery.

Without loss of generality, we delimit our analysis to those cases when an end device transmits UL packets to two BS at the same time. In this fashion, we filter the original dataset, obtaining a new subset with 17,146 samples. For example, a pair of samples from that subset represent an end device that concurrently sends UL traffic to BS 2 and BS 3, as shown in Figure 2.4. The default path for DL traffic depends on the BS with the highest RSSI, as we explained in Section 2.4.3. In our example, that default DL association is done via BS 2. Therefore, the decision to be made is whether DL packets are forwarded through the BS corresponding to the default path or not. In the latter case, the DL traffic would be transferred to BS 3. As mentioned in Section 2.2, our decision-making model is based on an MDP, so that the Network Server can make decisions on DL load balancing at each BS. We also model our MDP with some calculations based on data of the new subset.

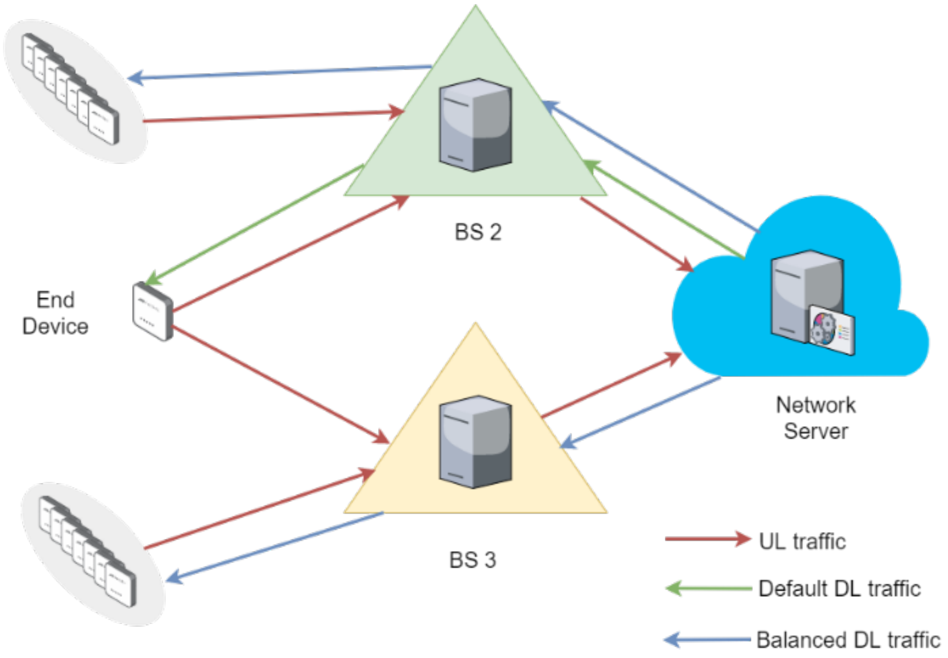


Figure 2.4. Example of a load-balancing decision to be made.

Generally speaking, an MDP is a sequential decision problem for an observable and stochastic environment with the Markovian property. In other words, MDPs are a fundamental formalism for sequential learning problems in stochastic domains, such as decision-theoretic planning and reinforcement learning [51]. A set of states s, a set of actions in each state $a(s)$, the transition probabilities among states $P(s'|s,a)$, and a reward function $R(s)$ comprise an MDP. Thus, a decision maker (also known as agent) must choose to perform an action when the process is in a singular state, based on a policy π, which is the decision solution given P and R [1]. For our scheme, we model an MDP with states corresponding to the number of BS. There are two actions to complete in each BS: to offload or not to offload its DL traffic, i.e., $a(s) = \{0, 1\}$. Two matrices describe the values of P for each action, defined as $\mathbf{P}_0$ when the decision is to not offload, and $\mathbf{P}_1$ to offload the BS.

We assume that any device is concurrently transmitting confirmed packets to two BS, which means that one of those BS must respond an ACK, i.e., a DL message. As we explained in Section 2.4.3, the default DL association for transmitting an ACK is between the BS with highest RSSI and the end device. Therefore, the decision that the Network Server has to make is whether the DL association remains with the default BS, $a(s) = 0$, or switches to the other one, $a(s) = 1$. Subsequently, the probability of being in state s and staying in that state if the decision is to not offload is $P(s' = s|s, a = 0) = 1$, and then $\mathbf{P}_0$ is defined as follows (2.9):

$$\mathbf{P}_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{ 2.9}$$

To calculate $\mathbf{P}_1$, the transition probabilities can be estimated from historical records [52]. Thus, we use the data samples to count the total number of device associations that each BS had and the shared associations between each pair of BS. Hence, the probabilities that the DL traffic is offloaded from a BS to another BS, $P(s'|s, a = 1)$, are given by (2.10):

$$\mathbf{P}_1 = \begin{bmatrix} 0 & \frac{\sum(\mathbf{x}_0 \cap \mathbf{x}_1)}{\sum \mathbf{x}_0} & \cdots & \frac{\sum(\mathbf{x}_0 \cap \mathbf{x}_6)}{\sum \mathbf{x}_0} \\ \frac{\sum(\mathbf{x}_1 \cap \mathbf{x}_0)}{\sum \mathbf{x}_1} & 0 & \cdots & \frac{\sum(\mathbf{x}_1 \cap \mathbf{x}_6)}{\sum \mathbf{x}_1} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\sum(\mathbf{x}_6 \cap \mathbf{x}_0)}{\sum \mathbf{x}_6} & \frac{\sum(\mathbf{x}_6 \cap \mathbf{x}_1)}{\sum \mathbf{x}_6} & \cdots & 0 \end{bmatrix} \quad (\ 2.10)$$

With respect to $R(s)$, we also estimate its values based on the historical data and the classifier results. Basically, we compute how busy a BS might be transmitting DL packets to define how "rewarding" that BS is. In this way, the more occupied a BS is, the higher its reward is for the offloading decision. This consideration is consistent with the fact that gateways utilization is taken into account to schedule DL traffic in TTN [46]. Then, the rewards vector for the MDP is calculated as follows (2.11):

$$\mathbf{R} = \sum_{n=1}^{N_A} \mathbf{r}_{k_n} \quad (\ 2.11)$$

where N_A is the total number of end devices requesting ACKs and $\mathbf{r}_k$ is the vector containing the obtained probabilities from the classifier.

It is also important to point out that we model our MDP with an indefinite horizon for the decision making, which means that there is no fixed time limit and that the optimal policy π^* is stationary [1]. Also, we consider a discount factor γ that describes the preference of the decision maker (in our case, the Network Server) for current rewards over future rewards. Accordingly, the utility of a state sequence is defined as (2.12):

$$U = \sum_k \gamma^k R(s_k) \quad (\ 2.12)$$

More importantly, we must find π^* for our MDP, which is an optimization problem to choose the action that maximizes the expected utility of the subsequent state (2.13):

$$\pi^*(s) = \text{argmax}_a \sum_{s'} P(s'|s,a)U(s') \tag{2.13}$$

There are several methods to solve this optimization problem. In this work, we assess the performance of two well-known algorithms: value iteration and policy iteration. On the one hand, the value iteration algorithm calculates the utility of each state and then iteratively uses the state utilities to select an optimal action in each state. On the other hand, the policy iteration algorithm alternates between the evaluation of the states utilities under the current policy (starting from some initial policy) and the improvement of the current policy with respect to the current utilities. Details about these and other algorithms can be found in [1], [52].

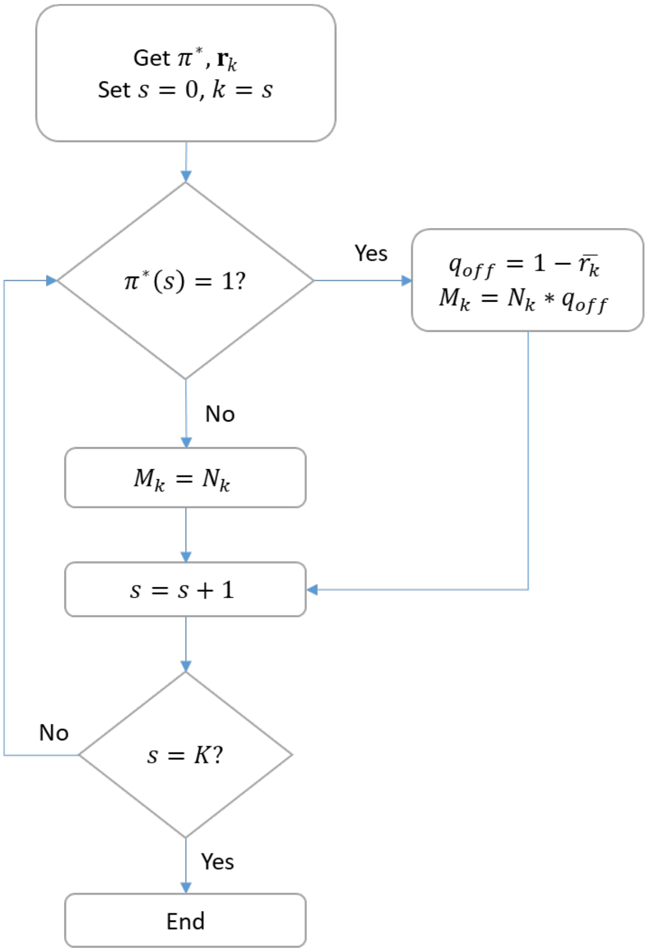


Figure 2.5. Algorithm for the traffic offloading decision.

Another point to consider is that we define the amount of DL traffic offloading based on the classifier outputs to avoid that any BS ends up with no packets to transmit. As a result, the quantity of end devices to be offloaded from a BS, that is M_k, depends not only on π^* but also on $\mathbf{r}_k$, as shown in Figure 2.5. In this flowchart, $\bar{r}_k$ is the mean value of vector $\mathbf{r}_k$, N_k is the number of devices associated with a specific BS, and K is total number of BS in the network.

2.5. Network Simulation Design

To simulate a system using our proposed scheme, we adapt some analytical models found in the literature for the simulation of LoRaWAN networks. As we assume that in the network all the devices are Class A, they use the uncoordinated transmission scheme ALOHA. The PDR in a network that employs pure ALOHA can be modeled based on a Poisson distribution [53], as follows (2.14):

$$\mathrm{PDR} = e^{-2N*T_{Packet}*\lambda} \tag{2.14}$$

where N is the number of devices in the network, T_{Packet} is the average airtime that takes transmitting a packet, and λ is the average packet arrival rate. However, this model does not take into account the retransmissions when devices request ACKs from the network. Therefore, the model is adapted to consider the retransmissions (2.15):

$$\mathrm{PDR}_A = e^{-2N_A*T_{Packet}*\lambda*p_{BS}} \tag{2.15}$$

N_A is the total number of end devices requesting ACKs, as described in Section 2.4.4, and the new term p_{BS} is the blocking probability of a BS due to the ACKs (DL traffic), given by (2.16):

$$p_{BS} = 1 - (1 - q_{BS})^{A_{Tx}} \tag{2.16}$$

where q_{BS} is the ratio between DL traffic and UL traffic in a BS and A_{Tx} is the number of retransmissions of a device before receiving an ACK. As the LoRaWAN standard specifies a maximum number of seven retransmissions and considering the original transmission as a retransmission, according to [54], we run our simulations with $A_{Tx} = 8$, which corresponds to the worst case. It is also important to highlight that, for the q_{BS} calculation, the DL traffic is either the default load or the balanced load at the BS, depending on the offloading decision.

Next, to simulate the PDR over all the BS in the network, we calculate the total PDR following the product form (2.17):

$$\mathrm{PDR}_{\mathrm{Total}} = \prod_{k=1}^{K} \mathrm{PDR}_{A_k} \tag{2.17}$$

where K is the total number of BS, that is $K = 7$ for our use case scenario.

In our simulations, each experiment represents an MDP. For each experiment, we take samples from the subset described in Section 2.4.4. In this manner, we conduct more realistic experiments by using actual data instead of synthetic data. An experiment consists of randomly selecting a pair of samples corresponding to an end device associated with two BS. Without loss of generality, we delimit our analysis to $N_{A_{\max}} = 5000$, starting with an experiment of 5 devices and increasing the number by 5 in each experiment. The main reason of this constrain is that most of the samples in the dataset correspond to end devices associated with one BS only. Then, in order to consider the end devices requesting DL traffic, we need to filter out those with a single association. In relation to T_{Packet}, we choose an airtime that is consistent with common LoRaWAN deployments like TTN. Consequently, we set $T_{Packet} = 1712.13$ ms, which is a robust packet airtime for those kinds of deployments, according to [55].

With respect to the energy cost of data delivery (ECD), we adapt the model for a dense LoRaWAN network presented in [56]. Thus, the ECD is given by (2.18):

$$\mathrm{ECD} = \alpha \frac{e^{N_A * \lambda * p_{BS} * L_{Pl}}}{L_{Pl}} \tag{2.18}$$

where α is a constant expressed in Joules and L_{Pl} is the size of messages payload. We assume that a typical Smart City IoT application transmits messages with a payload size of 20 bytes, on average. For both PDR and ECD models, Table 2.1 summarizes the parameters used in our simulations.

Table 2.1. Simulation parameters for the evaluation of Packet Delivery Ratio and Energy Cost of Data Delivery.

Parameter	**Description**	**Value**
γ	Discount factor for MDP	0.9
N_A	Number of end devices requesting ACK	{5, 10, 15, ⋯ 5000}
λ	Average packet arrival rate	0.25×10^{-4} packets/ms
T_{Packet}	Packet airtime	1712.13 ms
A_{Tx}	Number of retransmissions	8
α	Energy constant	0.4 J
L_{Pl}	Size of messages payload	20 B

2.6. Evaluation Results

We evaluate our method through computer simulations and, specifically, by running code written in Python 3. Some packages for data analysis and ML are used, such as pandas [57], scikit-learn [58], and MDPToolbox [59]. The simulations are run on a PC with

Ubuntu 16.04 64 bits, processor Intel® Core™ i3 CPU M 350 @ 2.27 GHz × 4, and RAM of 4 GB. Note that we decided not to use High Performance Computing systems, as we are aware that many private LoRaWAN deployments do not count on these sorts of resources. In the following subsections we present the numerical results of our simulations and discuss their implications.

2.6.1. PCA Patterns

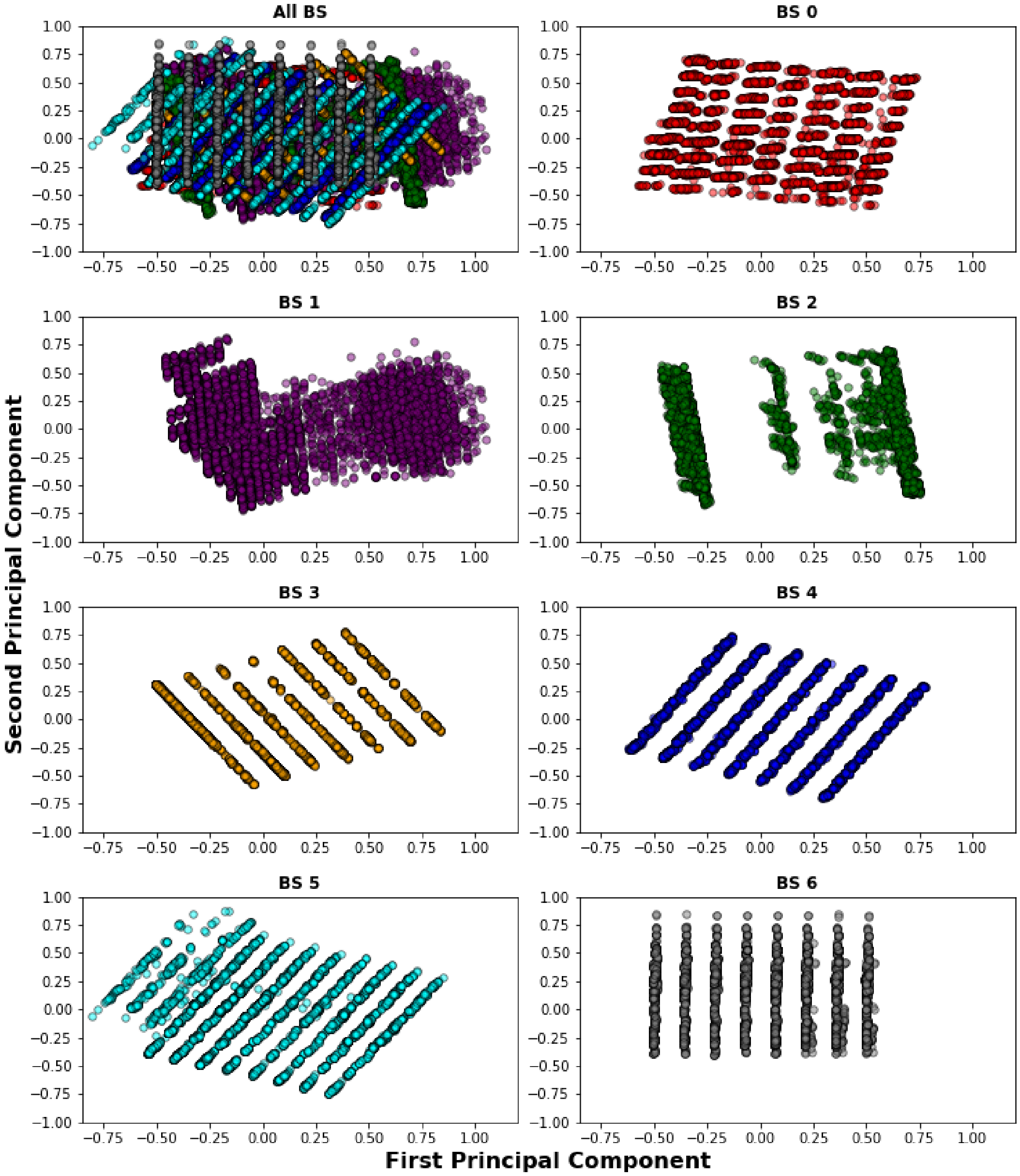


Figure 2.6. Discovered patterns for BS after projecting the first two principal components.

As explained in Section 2.4.2, we want to discover whether there is a characteristic pattern for each BS when the device association is biased by obviating the RSSI and SNR metrics. Therefore, the PCA analysis is performed taking into account the normalized values of the features: frequency, data rate, latitude, longitude, time of the day, and day of the week. To visualize the analysis, Figure 2.6 shows the pattern projected by the first two principal components for each BS. It is noticeable that each BS depicts a different pattern, which means that a device with particular values of the extracted features might be associated with specific BS through classification. In other words, we can predict the device's probability of having a specific DL path by biasing the signal-based variables.

2.6.2. Classifiers Outcomes

As we explained in Section 2.4.3, our goal with the classifier is to bias the default device-BS association based on signal strength measurements like RSSI. Then, the classifier is trained with features that represent the particular condition of the devices, excluding the signal-based variables. To evaluate the association biasing, we use data samples from the subset described in Section 2.4.4. In this manner, we select 8500 devices that simultaneously transmit UL packets to two BS. Assuming that the default association is given by the BS with the highest RSSI, an approximation of all BS coverage is shown in Figure 2.7a. As can be seen, this coverage mapping is comparable to that depicted in Figure 2.3. On the contrary, Figure 2.7b illustrates a coverage map estimate based on the classification results. In this case, the association of a device with the default BS is changed to the BS with the highest probability given by the classifier. It is noticeable that the proposed biasing method yields a CRE, as described in Section 2.2. For instance, the range of BS 1 and BS 3, which act as SBS, are virtually expanded after performing the association biasing.

To compare the performance of the probabilistic classifiers, we ran the training code 5000 times. Figure 2.8 shows the average classification accuracy and the average training times for each considered algorithm. The voting method is an ensemble classifier that combines the classification results from GNB and QDA.

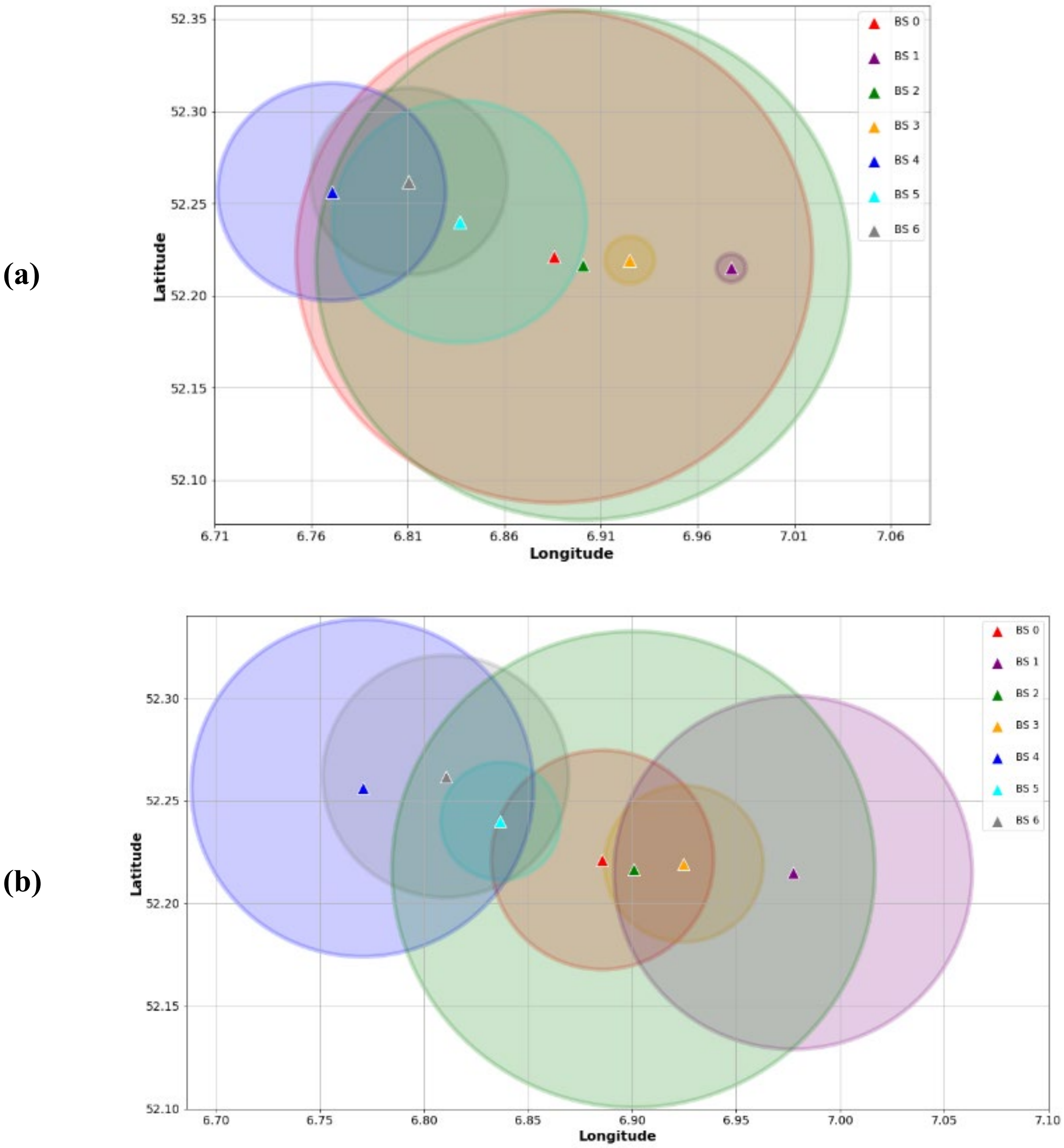


Figure 2.7. Device-BS association comparison: (a) estimated coverage based on RSSI; (b) CRE based on biased association.

As can be seen, the most accurate classifier is ET, however, this algorithm also employs the third longest training time. In contrast, our intention with the voting classifier is to evaluate any accuracy enhancement given by the combination of the two fastest algorithms, i.e., GNB and QDA. Although the accuracy of the voting algorithm is slightly above the QDA’s score, the total training time is approximately the summation of their individual training times. For these reasons, we finally use the ET algorithm outcomes as inputs for the decision-making model.

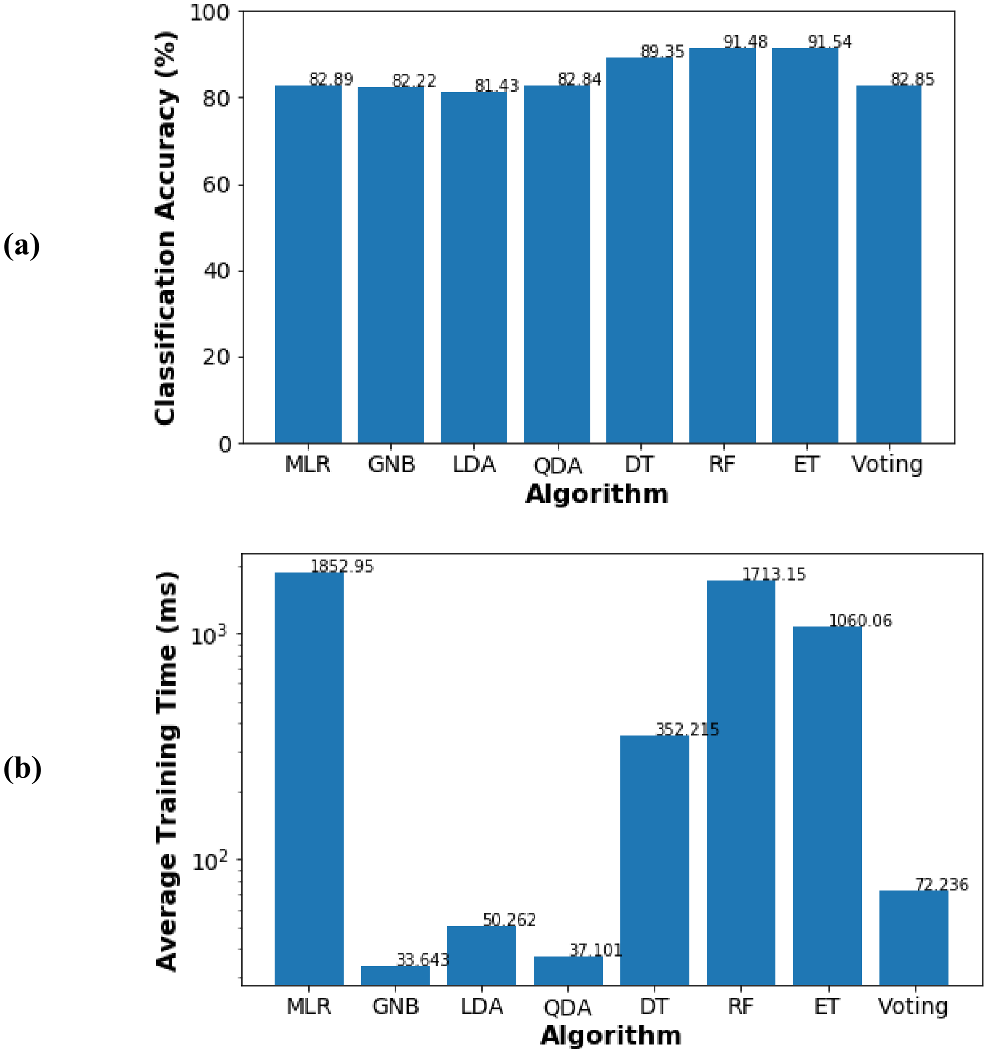


Figure 2.8. Classifiers performance comparison: (a) average classification accuracy; (b) average training time.

2.6.3. Network PDR Improvement

In this subsection we present the obtained results when the network is simulated with the parameters specified in Section 2.5. We first compare the PDR improvement achieved with our proposed scheme through two simulation setups: MDP with and without the classifier. The objective is to determine if the combination of the classification method and the modeled MDP really makes a difference compared to the MDP working alone. Note that, to compare the MDP results without the classifier, the reward vector **R** is found by counting the number of default DL associations. In this way, one simulation setup is based on the association biasing given by the classifier's predictions (as described in Section 2.4.4) and the other setup relies on the RSSI-based association.

Figure 2.9 depicts the resulting graphs of the system simulation in terms of PDR. As can be seen, the proposed scheme performs better when the outcomes from the classifier are taken into account, particularly in the circumstances when many devices are requesting DL traffic (note that the MDP-only load balancing method outperforms the combined method just when the number of devices is small, i.e., between 0 and 300 devices, roughly). However, there is a trade-off between the PDR improvement and the computational time, Figure 2.10. We point out that in this comparison we only consider the classifier's prediction time, in other words, we do not include its training time, as we assume that the Network Server has previously trained the model. It is noticeable that when the proposed scheme uses the association biasing based on the predicted classes, the MDP needs more time to make a decision on load balancing. It is also important to highlight that the graph show some peaks, which means that the iteration algorithm employed more iterations to find π^*. Because of the stochastic nature of the samples, the algorithm might have dealt with tough values to determine π^*. However, we can see that in those cases, although more time was needed, the goal of improving the PDR was achieved.

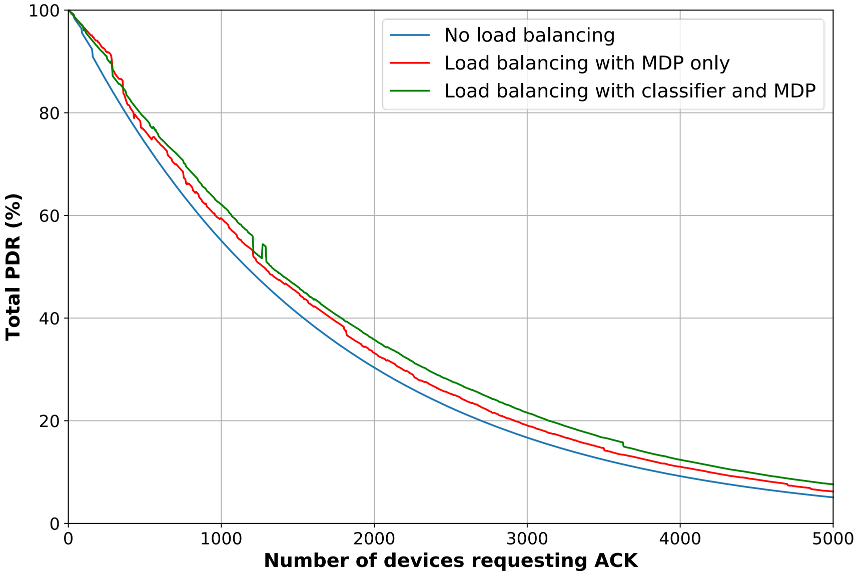


Figure 2.9. Network PDR improvement based on proposed scheme.

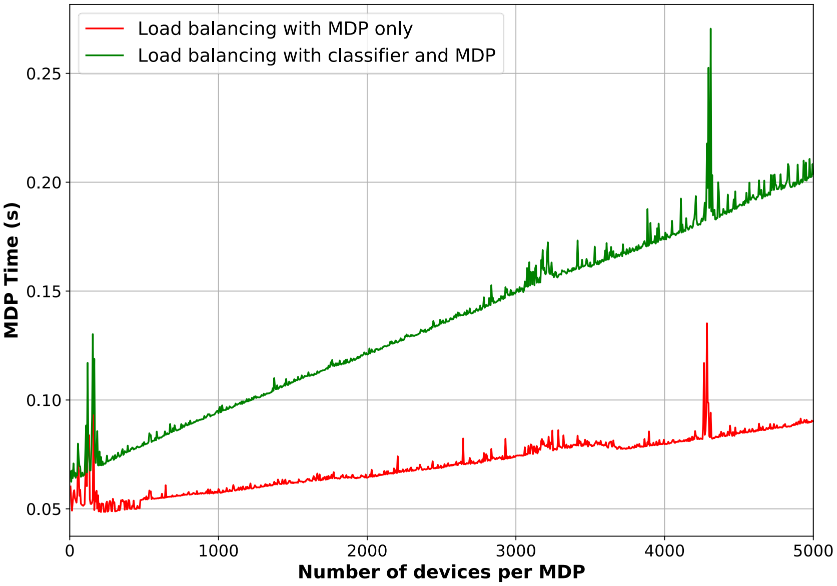


Figure 2.10. Computational time comparison for the MDPs.

In terms of improvement percentages, we find that the PDR increases by 13.11%, on average, and up to 26.8% without the classifier. Similarly, the PDR rises by 23.74%, on average, and up to 49.98% when the classifier results are incorporated in the decision-making model. In contrast, the average decision time is 89.33% higher for the latter case, reaching a maximum of 0.27 s. However, we highlight that the decision process is run on the Network Server which is supposed to have enough resources to deal with this trade-off and take advantage of a better PDR for the whole network.

Additionally, as mentioned in Section 2.4.4, we compare the computational time of both value iteration and policy iteration algorithms to solve the MDPs. The measured average decision times are 70 ms and 95 ms for the policy iteration and the value iteration methods, respectively, after running the experiments with the MDP-only simulation setup. This fact reveals that the policy iteration method is about 26% faster than the value iteration method to find the optimal policy of our load balancing decision model. That is why we used the policy iteration algorithm for the comparison described in Figure 2.10.

2.6.4. Network ECD Reduction

In relation to the ECD, we also compare the results of the MDPs with and without the association biasing. Figure 2.11 depicts the normalized ECD. Similar to the PDR evaluation results, our proposed scheme yields an ECD reduction of 8.1%, on average, and up to 13.36% when the classification method is ignored. Conversely, a maximum reduction of 19.1% and an average ECD reduction of 12.04% are achieved when the biasing method, based on the classifier, is included in the load balancing model.

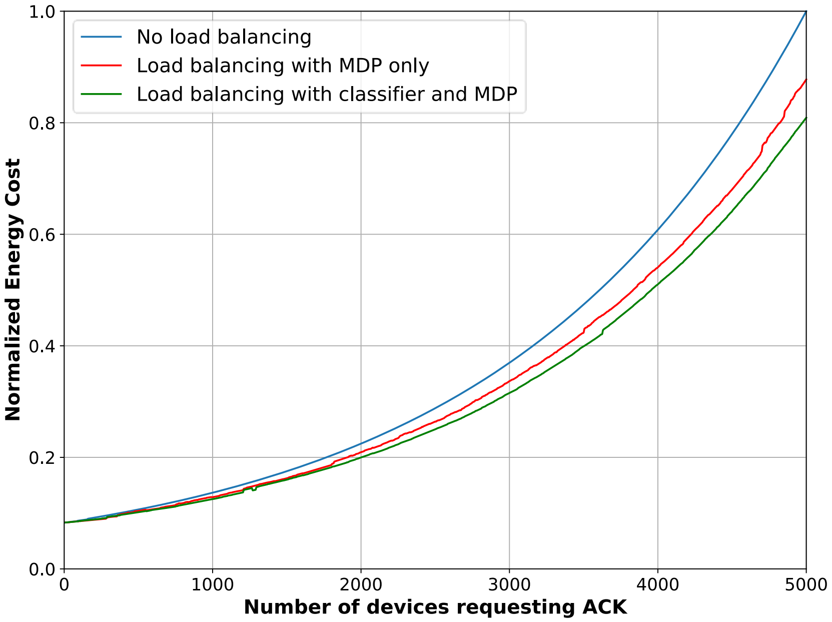


Figure 2.11. Network EDC reduction based on proposed scheme.

2.7. Summary

With the dramatic increase of connected devices, the Internet of things (IoT) paradigm has become an important solution in supporting dense scenarios such as smart cities. The concept of heterogeneous networks (HetNets) has emerged as a viable solution to improving the capacity of cellular networks in such scenarios. However, achieving optimal load balancing is not trivial due to the complexity and dynamics in HetNets. For this reason, we propose a load balancing scheme based on machine learning techniques that uses both unsupervised and supervised methods, as well as a Markov Decision Process (MDP). As a use case, we apply our scheme to enhance the capabilities of an urban IoT network operating under the LoRaWAN standard. The simulation results show that the packet delivery ratio (PDR) is increased when our scheme is utilized in an unbalanced network and, consequently, the energy cost of data delivery is reduced. Furthermore, we demonstrate that better outcomes are attained when some techniques are combined, achieving a PDR improvement of up to about 50% and reducing the energy cost by nearly 20% in a multicell scenario with 5000 devices requesting downlink traffic.

Chapter 3

Intelligent Active Queue Management

3.1. Motivation

Thanks to the proliferation of smart devices and the paradigm of Internet of Things (IoT), the demand for connections to the Internet is dramatically growing. As a response, Internet Service Providers (ISPs) are focused on improving the performance of their networks and connections to the Internet. However, engineers and researchers are trying to address this challenge by solving the traditional networks' congestion problems. On the one hand, congestion avoidance mechanisms in TCP have been part of the solution and essential for the massive adoption of the World Wide Web. On the other hand, due to the bottlenecks along the paths, buffers have been deployed to avoid packet loss when packets arrive at faster rate than can the links. Nevertheless, excessive buffering leads to increasing delays, as packets have to stay longer in the queues, and causing a phenomenon known as bufferbloat [60]. Network devices tackle this effect through Active Queue Management (AQM) techniques, which aim to avoid the buffer's overflow by dropping or marking the packets before the buffer fills completely. A variety of AQM schemes has been proposed, including the classical Random Early Detection (RED) algorithm [61], the Controlling Queue Delay (CoDel) [62], and newer ones such as the Proportional Integral controller Enhanced (PIE) [63] and the Flow Queue CoDel (FQ-CoDel) [64]. Despite the advantages of AQM techniques, they are not widely adopted in ISPs' network devices for the following reasons: first, some AQM mechanisms have parameters that might be difficult to tune in very dynamic environments. Second, routers and switches with more memory available in the market have created the misconception that the larger the buffers, the better.

The main advantage of dropping packets with AQM rather than with tail-drop queues, *i.e.* buffers with no AQM, is to avoid the unnecessary global synchronization of flows when a queue overflows. Consequently, network devices drop more packets when no AQM scheme is in use and the network throughput is deteriorated. In contrast, an AQM method can decide to either drop or mark packets when the network experiences incipient congestion. The process of marking packets instead of dropping them is known as Explicit Congestion Notification (ECN). The employment of ECN can reduce the packet loss and latency of Internet connections, among other benefits such as improving throughput, reducing probability of retransmission timeout expiry, and reducing the head-of-line blocking [65]. Moreover, the importance of ECN relies on its fact of making incipient congestion visible, by exposing the presence of congestion on a path to network and transport layers. The data containing ECN-marked packets can be exploited to learn some characteristics such as the level of congestion of a network operator and the behavior of TCP protocols or applications, for instance. For these reasons, the deployment of new ECN-capable end systems and the necessity of reducing queuing delay in modern networks have motivated the interest in ECN [66]. Indeed, IETF has published a significant number of RFC documents regarding ECN, which indicates strong level of interests from industry and academia.

ECN is specified in the RFC3168 [67], which defines four codepoints through two bits in the IP header, to indicate whether a transport protocol supports ECN and if there is congestion experienced (CE). This IETF recommendation also specifies two flags in the TCP header to signal ECN: the ECN-Echo (ECE) and the Congestion Window Reduced (CWR). Then, if the AQM algorithm in any router along the path determines that there is congestion, the router marks the packets with the CE code to indicate to the receiver that the network has experienced congestion. Once the CE-marked packet arrives at the receiver, it echoes back a packet to the sender with the ECE flag set in the TCP header to notify that congestion was experienced along the path. Consequently, the sender reduces the data transmission rate and sends the next TCP segment to the receiver with the CWR flag set. It is important to highlight that TCP also responds to non-explicit congestion indication produced by tail-drop queues or AQM dropping. How TCP performs those actions depends on the congestion control mechanisms on the transport layer and their details are out of the scope of this work. However, it is evident that the utilization of ECN mitigates the need for packet retransmission and, consequently, avoids the excessive delays due to retransmissions after packet losses. In addition, without ECN it is not possible to determine if the packets are lost because of congestion or poor link quality. Finally, we point out the rest-of-path congestion concept introduced in the Congestion Exposure (ConEx) mechanism, which to some extent has inspired our work. Although proposed several years ago, the implementation of ConEx is not widely deployed, as it needs modifications to the TCP protocol at the sender side [68].

Accordingly, in this work we propose an intelligent use of the standardized ECN mechanism for existing AQM solutions. We build our method on Machine Learning techniques for the exploitation of ECN. The method consists of two main parts: a congestion predictor and a dynamic parameter tuner. The latter applies a Reinforcement Learning (RL) technique to balance the delay and throughput by adaptively setting the AQM parameters. The congestion predictor is a Neural Network (NN) that forecasts if there will be congestion on the rest-of-path. Our main goal is to propose a scheme that is fully compatible with existing TCP congestion control mechanisms and already deployed AQM techniques. Although previous works have used Machine Learning techniques to solve problems regarding AQM, to the best of our knowledge, none of them exploits ECN to improve the AQM mechanisms. For example, authors in [69] compare several AQM techniques based on NN with conventional AQM techniques. Through simulations, the authors show that the studied NN-based methods converge faster than the traditional techniques. Similarly, Bisoy and Pattnaik propose in [70] an AQM controller based on feed-forward NN, which stabilizes the queue length by learning the traffic patterns. Also, on the basis of RL, Bouacida and Shihada present in [71] the LearnQueue method, which focuses on the operation in wireless networks. Authors model their solution by adapting the Q-learning algorithm to control the buffer size. By means of unsupervised learning techniques, authors in [72] propose a cognitive algorithm to detect and penalize misbehaving ECN-enabled connections. Although this problem and the employed techniques differ from ours, we find some similarities in terms of exploiting the TCP connection data and the implementation on top of existing AQM mechanisms.

3.2. Intelligent AQM Design

As we mentioned in the Introduction, our goal is to enhance the performance that current AQM techniques provide at bottlenecks. We have explained how the ECN can reduce the connections' latency when enabled in the AQM along a path. However, ECN is not currently exploited to estimate the congestion ahead and dynamically adjust the AQM parameters in a router. Our hypothesis is that TCP connections can have a better performance if AQM schemes are tuned based on the specific network conditions. Yet, this is a non-trivial problem due to the complexity of IP networks. Consequently, we propose an intelligent method for improving existing AQM that learns from the experience and ECN feedback of a changing network. Our method is meant to be implemented on edge routers for two main reasons: first, edge routers are more prone to experience congestion than core routers, due to the bottleneck link between the access network and the backbone. Second, our mechanism uses traffic data in the downstream direction, which may take different paths in the core network. Despite these reasons, our solution can be deployed in core network devices even if ECN feedback is not completely obtained. The overall scenario for our stated problem is shown in Figure 3.1, which is a valid topology for end points connected through a shared bottleneck link [73]. It is also important to highlight that ECN is not a perfect mechanism for congestion control. If an AQM decides to mark every packet with incipient congestion regardless the status of the queue, the AQM could produce a harmful effect. That is why we argue that a right and dynamic setting of the AQM parameters is pertinent. Moreover, we point out the potential application of Machine Learning techniques for this purpose.

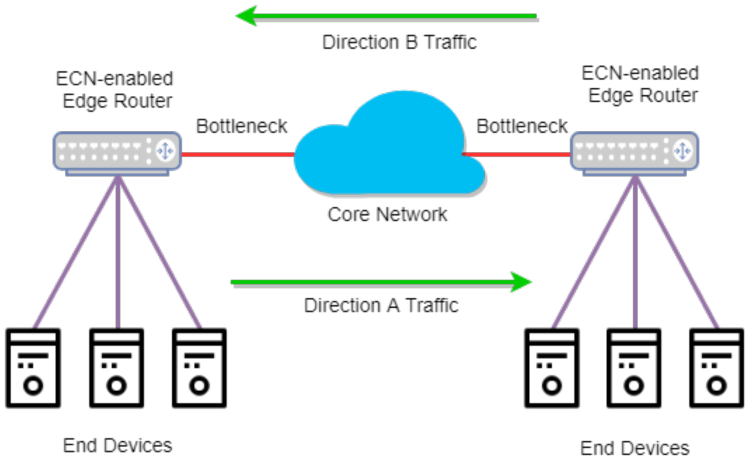


Figure 3.1. Scenario for our stated problem. Edge routers aggregate end devices and connect to the core network through bottleneck links.

3.2.1. Congestion Predictor

To predict the congestion, we take advantage of the ECE flag available in the TCP header of the packets in direction B without considering the ones involved in the ECN negotiation, as those packets indicate the setting of ECN-capable TCP sessions rather than congestion or response to congestion [67]. We model the congestion prediction as a time-series problem. The core of the congestion predictor is a Long Short-Term Memory (LSTM), which is a Recurrent Neural Network (RNN) architecture with memory blocks in the hidden layers. The memory blocks have multiplicative gates that allow storing and accessing information over long periods. In this way, the vanishing gradient problem of the RNN is mitigated in the LSTM, since the gradient information is preserved over time. For this reason, LSTMs have been successfully applied to address real-world sequential and time-series problems [74]. The inputs consist of both the current sample and the previous observed sample, such that output at time step t -1 affects the output at time step t. Each neuron has a feedback loop that returns the current output as an input for the next step. This structure makes LSTMs an effective tool for prediction, especially in those cases where there is no previous knowledge about the extent of the time dependencies.

The inputs of our LSTM-based congestion predictor are denoted as a sample vector with the number of ECE-marked packets arriving at time intervals of 100 ms. This value corresponds to the typical assumption for the Round-Trip Time (RTT) in IP networks. Additionally, we rearrange that vector as an input matrix **X** corresponding to ten time steps and an output vector **y** of one time step, such that:

$$\mathbf{X} = \begin{bmatrix} x_{t_0} & x_{t_1} & \cdots & x_{t_9} \\ x_{t_1} & x_{t_2} & \cdots & x_{t_{10}} \\ \vdots & \vdots & \ddots & \vdots \\ x_{t_{N-10}} & x_{t_{N-9}} & \cdots & x_{t_{N-1}} \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} x_{t_{10}} \\ x_{t_{11}} \\ \vdots \\ x_{t_N} \end{bmatrix} \tag{3.1}$$

where x_{t_i} is the quantity of ECE-marked packets in the time interval i and N is the total number of samples. The rationale behind rearranging the samples in ten time steps is to improve the performance of the predictive model by having additional context. In this way, the estimation of arriving ECE-marked packets contemplates more prior observations.

For the design and training of the LSTM, we assume that the data are gathered in a ten-minute period, which is reasonable due to the dynamics of Internet networks. Consequently, there would be a dataset with 6000 samples, corresponding to the number of intervals of 100 ms in ten minutes. In addition, we consider an LSTM with three hidden layers: the employment of a low number of layers for LSTM has been well studied in the literature and, based on our own experimentation, we were able to confirm that three layers are enough for making accurate predictions, as presented in [75]. Also, we use the approximation formula proposed in [76] to determine the number of neurons per layer, as follows:

$$N_n = \left(N_{in} + \sqrt{N}\right)/L \tag{3.2}$$

where N_{in} is the number of inputs, N is the number of samples, and L is the quantity of hidden layers. Then, $N_n \approx 30$ neurons per hidden layer. Although this formula was empirically determined for time-series forecasting using Feed-Forward Neural Networks, our experimentation show that it also works well for RNNs. Finally, we take into account a dropout regularization of 20%, so that the model does not overfit and yields more generalized weights after training.

3.2.2. Q-learning based AQM Parameter Tuner

In general, the parameters of AQM algorithms are set to values that yield a reasonable performance for the typical network conditions. However, AQM mechanisms are expected to allow parameters adjustment depending on the specific characteristics of a network and their interactions with other network tasks over time [77]. Consequently, we embrace the idea of adjusting AQM parameters according to the network's changing circumstances, so that the performance is dynamically improved, as well. Nevertheless, the achievement of this goal can end up in a very complex job. For this reason, we propose a mechanism that adaptively tunes the parameters of the AQM in use as an RL-aided decision process.

We model the dynamic AQM parameter-tuning problem as a Markov Decision Process (MDP). Previous works have successfully modeled complex decision-making problems in networks through MDPs, [78]. For this intelligent method, the decision process is based on the inferred rest-of-path congestion, *i.e.* the output of our congestion predictor described in Section 3.2.1. In this way, we define the states S as a set of discrete levels of congestion that the flows will be likely to experience along the path, the set of actions A comprises specific values of the target parameter, and the reward R depends on the power function of the connection, which is defined as the throughput-to-RTT ratio. In our environment, the edge router acts as the agent that makes the decisions and, therefore, no extra intelligence is needed at the end devices. The idea behind using the predicted rest-of-path congestion is to proactively tune the AQM at the edge router. Consequently, our method can adjust the target parameter so that more packets are dropped instead of being marked, as they will be likely dropped ahead. On the other hand, if low congestion is forecasted ahead, the AQM will mark more packets based only on its own experienced congestion.

Nevertheless, finding the appropriate target for the balance between dropping/marking packets is a non-trivial problem and that is why we use RL. In other words, we model our problem as an MDP with the objective of finding an optimal behavior that maximizes the throughput-to-RTT ratio. To do so, we utilize the Q-learning algorithm [79], which defines a function $Q(S, A)$ representing the quality of a certain action in a given state and that is defined by:

$$Q(S, A) := Q(S, A) + \alpha \left[R + \gamma \max_a Q(S', a) - Q(S, A)\right] \tag{3.3}$$

where $a \in A$, $\alpha \in [0,1]$ is the learning rate, and the discount factor $\gamma \in [0,1]$ describes the preference of the agent for current rewards over future rewards. This equation characterizes the maximum future reward of present state and action in terms of immediate reward and maximum future reward for the next state S'. In this manner, the Q-learning algorithm iteratively approximates the function $Q(S, A)$.

More specifically, we model our AQM parameter tuner considering the current states as the observed levels of congestion, *i.e.* the ECE-marked packets arriving at the router in direction B, and the rest-of-path congestion prediction in direction A as possible next states. Both current and next states are discretized to delimit the complexity of the environment. On the other hand, the actions are a set of predefined values for the target parameter of the specific AQM in use.

3.3. Evaluation Methodology and Results

In this section, we provide the details about the experimentation setup for the evaluation of our proposed solution. We first explain the preliminary experiments conducted to show the feasibility of our method as a whole, by studying the basis of each component separately. Later, we evaluate the performance of our intelligent AQM scheme comparing its operation to the behavior of conventional AQM. For our experimentation, we use the Mininet network emulator and the queue disciplines available in the Linux kernel. In this way, we validate the potential deployment of our solution in real network scenarios.

3.3.1. Effects of Tuning AQM Parameters

With respect to the AQM parameter tuner, in this work we evaluate our proposal using CoDel [62] and FQ-CoDel [64]. Therefore, the target parameter to tune is the acceptable standing/persistent queue delay. Both in CoDel and FQ-CoDel, the minimum local queue delay is measured and compared with the value of the acceptable queue delay given as a target. To ensure that the minimum value does not become stale, the delay is measured within the most recent interval and, typically, the target delay is 5% of that interval. In this way, when the queue delay exceeds the target, a packet is dropped and a control law sets the next drop time. When the queue delay goes below target, the controller stops dropping [62].

To show the influence of changing the target parameter in both RTT and throughput metrics, we conducted some preliminary experiments by implementing a topology like the one depicted in Figure 3.1. In the emulation scenario, the edge router on the left (R1) performs the AQM control and has 20 hosts, *i.e.* hosts B, connected to it. On the other side, 20 hosts connect to the right edge router (R2): these are hosts A. There are also a pair of monitor hosts, and one of them actively logs the measured RTT (mRTT) and throughput by means of sending probe packets to the other one. Note that for this experiment we consider a propagation delay of 20 ms and a bandwidth of 200 Mbps between hosts B and R1. Conversely, there is no propagation delay from R1 to R2 and, to emulate the path bottleneck, the link between the two routers has a bandwidth of 20 Mbps. The links between R2 and the hosts A have a bandwidth of 100 Mbps and no propagation delay. In addition, all hosts are ECN-enabled and each pair of hosts AB generates TCP traffic, mainly in direction A. In this work, we conduct our experimentation only with CUBIC, the default TCP congestion control in Linux.

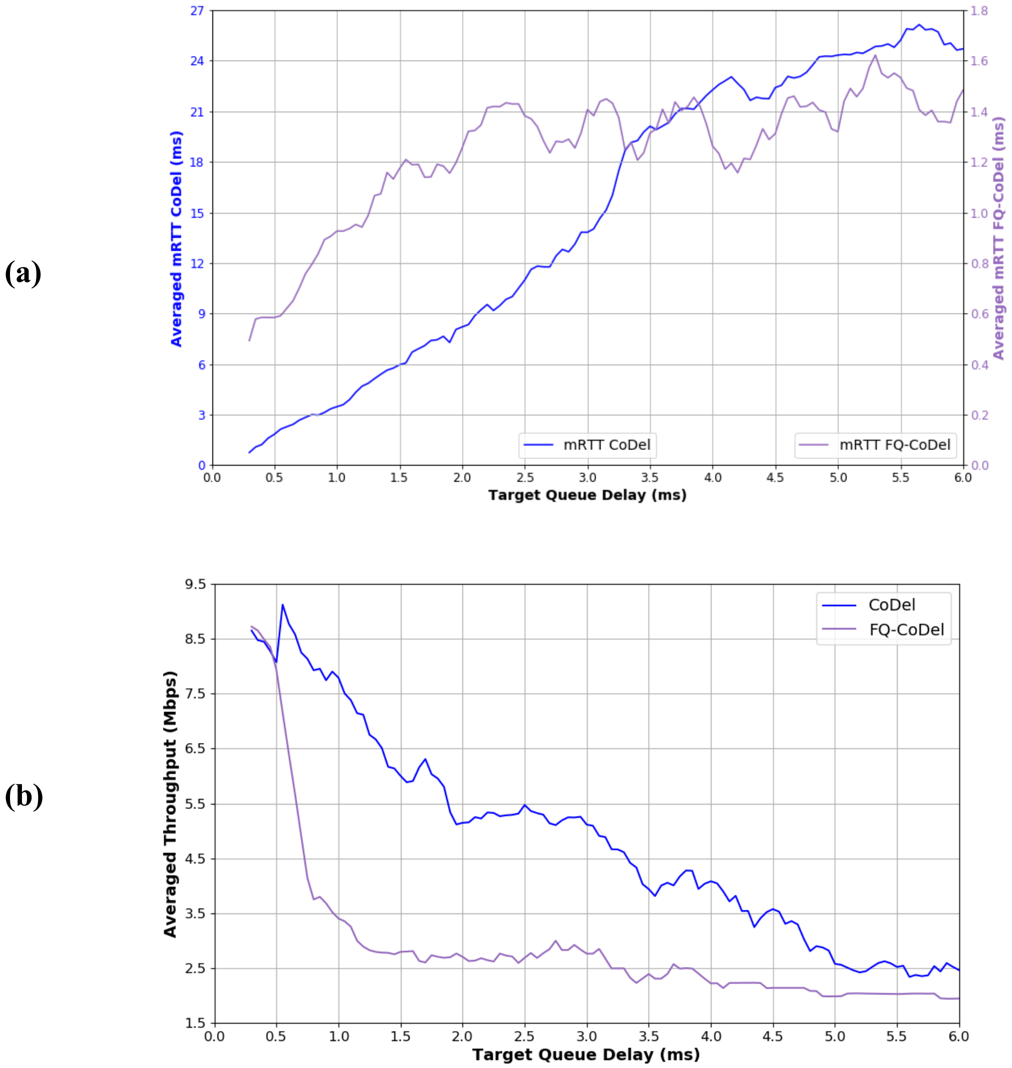


Figure 3.2. Effects of varying the target parameter in CoDel and FQ-CoDel algorithms on: a) Averaged mRTT. b) Averaged throughput.

The experiment consists of modifying the target and interval parameters of CoDel and FQ-CoDel in R1, while data are constantly and simultaneously transferred from the hosts B to hosts A. Therefore, we set CoDel and FQ-CoDel in R1 with target values from 50 µs to 6 ms and intervals from 1 ms to 120 ms, respectively. We left the other parameters as default, except the hard limit on the queue size, which we set to 1000 packets. This a configurable parameter set by the system administrator and the assumption of this value is based on the fact that small buffer sizes in backbone routers are sufficient for many networks and recommended for overall scalability [80], [81]. In addition, we were able to determine that the assumed hard limit was enough for all the queues of our emulation setting, i.e. there was no overflows at any buffer.

Figure 3.2 shows the resulting average mRTT and throughput for both queue disciplines in this experiment. Note that Figure 3.2a has two different scales for the y-axis, since the mRTT is significantly longer for CoDel. As can be seen, although the target parameter of these AQM algorithms is meant to operate unchangeably, there is a noticeable effect when the target parameter varies. The lower the target queue delay, the more dropped packets, since not all packets can be ECN-marked when the router experiences congestion. Consequently, RTT is low and throughput is high when low target delay is configured, Figure 3.2b. In other words, as the target parameter increases, the AQM mechanism produces more ECN-marked packets and drops less. This is consistent with our solution formulation explained in Section 3.2.2.

3.3.2. Transferring the Predictor Model

As an initial training and test for our congestion predictor, we use the data from a backbone Internet link of an ISP collected by the Center for Applied Internet Data Analysis (CAIDA). The CAIDA's monitors collect packet headers at large peering points and a wide variety of research projects has used its anonymized traces [82]. Specifically, we use the data from the collection monitor that is connected to an OC192 backbone link (9953 Mbps) of a Tier 1 ISP, between New York, US, and Sao Paulo, Brazil. We use this dataset as valid data for an edge router, according to previous works cited at CAIDA's website, in which those data have been used similarly. In particular, we chose to analyze the data from December 20, 2018.

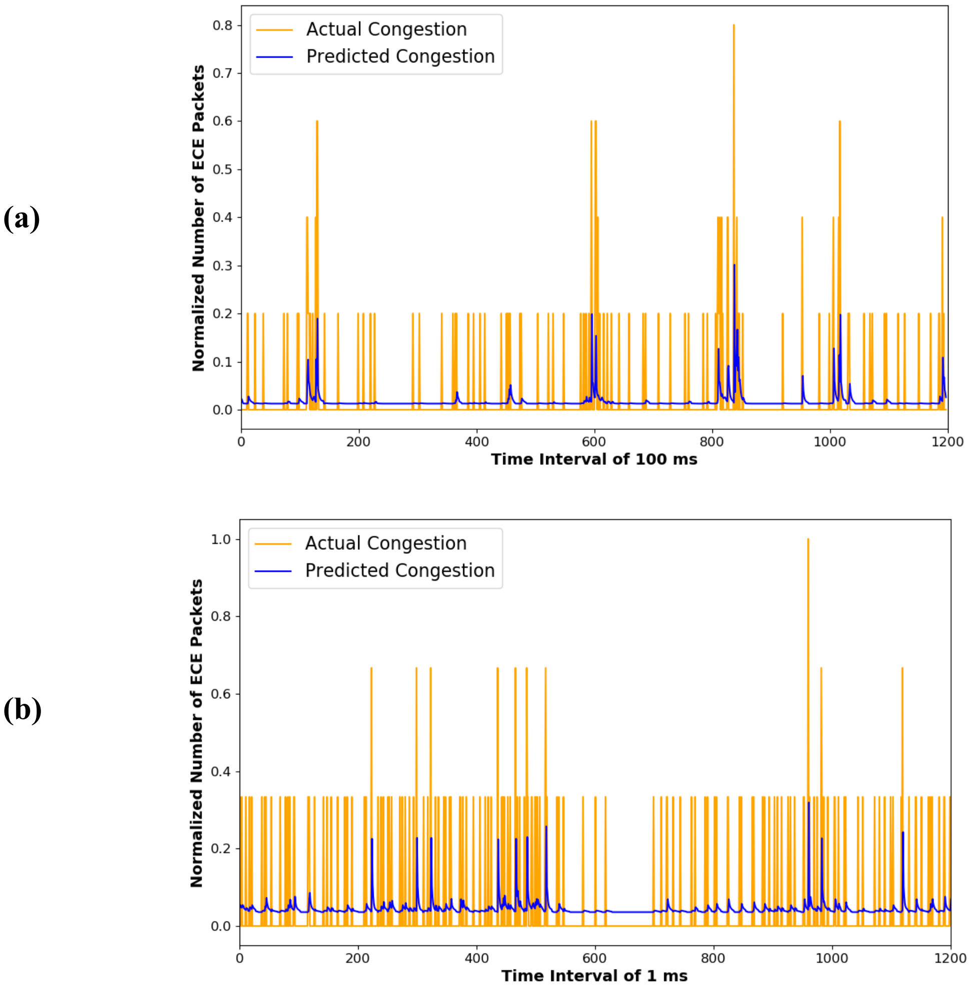


Figure 3.3. Actual and predicted congestion obtained after: a) Pre-training over 100 epochs, using the CAIDA's dataset and time intervals of 100 ms. b) Re-training in one epoch, based on network emulation data in time intervals of 1 ms.

We perform the pre-training for the congestion predictor with data containing ECE-marked packets sent from New York to Sao Paulo, as we found that there are more ECE-marked packets in direction B than in direction A. According to the assumptions explained in Section 3.2.1, we use the trace data in the ten-minute period with the highest number of ECE-marked packets that are not part of the ECN negotiation, that is from 8:00 to 8:10 EST. The traces show that, in this period, there were 402 different source IPv4 addresses sending ECE-marked packets to 315 destination hosts. We split the dataset into a training subset, corresponding to 80% of samples, and a test subset with 20% of samples. After 100 epochs of training, we test the model by making predictions with samples from the normalized subsets. We obtain a Root Mean Squared Error (RMSE) score of 0.08 and a Mean Absolute Error (MAE) score of 0.04 for the test subset. Similarly, we get an RMSE of 0.07 and a MAE of 0.03 for the training subset.

Figure 3.3 shows the actual normalized number of ECE-marked packets arriving at the router in direction B and the prediction over the test subset. As can be seen, the white spaces in the graph mean consecutive time intervals with no congestion, *i.e.* no ECE-marked packets at the router. On the other hand, the transients depict the time intervals in which congestion was experienced. Note that the levels of congestion correspond to the number of ECE-marked packets that arrive within an interval. In this way, we model the predictor to estimate whether there will be a significant level of congestion describing a transient in the number of ECE packets within the next time interval. Figure 3.3a illustrates how the resulting prediction captures the intervals when the levels of congestion ahead show those transients.

Hence, we use the pre-trained LSTM model to accelerate the congestion estimation in our method. As the network conditions change, our method updates the predictor by re-training it with new data. However, this re-training process is much faster, as the LSTM updates in just one epoch, which takes about four seconds in our emulation environment. To see how the pre-trained congestion predictor behaves in a new environment, we run an experiment with the topology described in Section 3.3.1. Moreover, to stress the network and make it more stochastic, we set random values of bandwidth and propagation delays on the links between hosts and routers. Likewise, each host B starts its transmission at a random time. The link bandwidth between R1 and R2 is the only non-random value fixed at 10 Mbps. Also, FQ-CoDel is the AQM method in this experiment with its default target delay and interval values, which are 5 ms and 100 ms, respectively.

In relation to the re-train process, we update the model with data gathered in six seconds. The rational behind this assumption is that network traffic changes very fast and so does its data. This situation can produce a model drift, which means that the relationship between the target variable and the input variables changes with time. Due to this drift, the model may become unstable and start making erroneous predictions over time [83]. It is then evident that a model drift might happen more often in our scenario than in other non-networking environments.

Consequently, as we designed the congestion predictor for 6000 intervals (see Section 3.2.1), we need to reduce the value of each time interval for the updates. Then, in this case, we re-train the LSTM with data in time intervals of 1 ms. After one update, the obtained values of RMSE are 0.09 and 0.13 for training and test subsets, respectively. In the same way, the resulting values of MAE are 0.04 for training and 0.06 for test. These scores show that our model can make predictions in the new network with a significant approximation and without the need for training the model from scratch. Figure 3.3b depicts the congestion prediction results in the described network. Note that we scale by two times the graph corresponding to the prediction, *i.e.* the blue plot, for clarity of the comparison. Again, rather than the exact number of ECE-marked packets, we want to predict the transients of congestion level ahead.

3.3.3. Performance Evaluation of the Intelligent AQM

In this subsection we elaborate more about the experiments that we conducted to show the job of our proposed method as a whole. In Section 3.2.2, we briefly described how the congestion predictor integrates with the AQM parameter tuner. We evaluate the MDP for this problem considering 100 levels of congestion as current or next states. The observed congestion corresponds to the current state and the predicted congestion is the next state. To determine their levels, we keep the maximum observed and predicted values as reference for the discretization. We also delimit the actions to 100 values, which in this case are the target delay of CoDel and FQ-CoDel. In this way, the possible actions are a set of values from 50 μs to 5 ms in steps of 50 μs. As we explained in Section 3.3.1, we modify two parameters at the same time: the target delay and the interval. Thus, the experiments are more consistent, as these two parameters are tightly related. Again, the hard limit buffer size is set to 1000 packets and the TCP congestion control is CUBIC. The starting values for the target and the interval parameters are the default ones in the Linux kernel: 5 ms and 100 ms, respectively. For this evaluation, R1 performs the intelligent AQM while R2 needs only to be configured as ECN-enabled or as a regular router that does not wipe CE-marked IP packets.

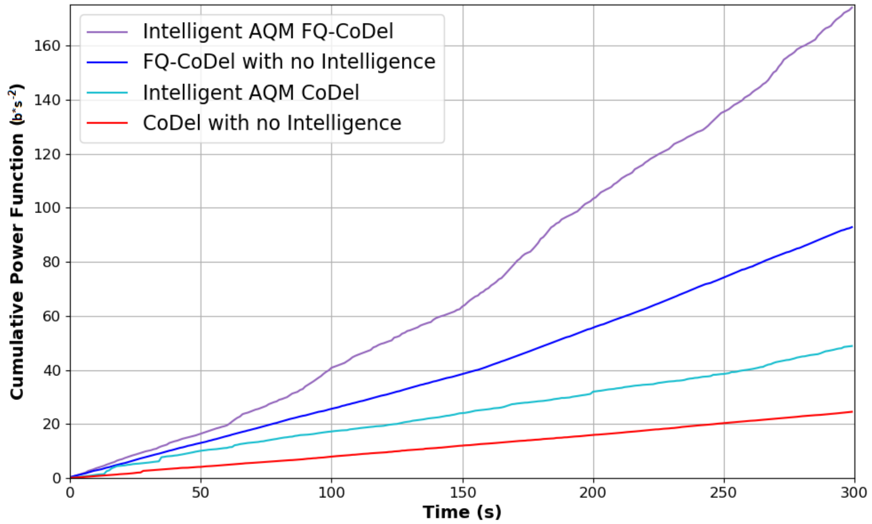


Figure 3.4. Cumulative power of the connection measured during the experiments in the emulation environment. The intelligent method is applied to CoDel and FQ-CoDel. All schemes utilize ECN

Figure 3.4 shows the results comparison when our intelligent method is applied to CoDel and FQ-CoDel, in terms of the cumulative power function. Note that these AQM schemes have static target parameters set to their default values when no intelligence is dynamically adapting them. As any other RL-based solution, the basic idea is to have an agent, *i.e.* the edge router in our problem, making decisions and getting feedback from the environment to calculate the rewards. To achieve so, we constantly capture the ECE-marked packets arriving at the router in direction B. Every second, the agent predicts the congestion of the rest-of-path in direction A. As the agent does not know what action to take at the beginning, there is an initial stage of exploration, which depends on the parameter ε. The value of this parameter determines if the Q-learning algorithm prefers to explore rather than exploit the historical data. In our experiments, we set $\varepsilon = 0.5$ so that the algorithm does not explore too greedily. After taking an action, either by randomly exploring or by extracting Q-values, the monitoring hosts measure the mRTT and throughput with active probes. We use these measures to calculate the power of the connection, which is our reward function. Once the rewards are known, the algorithm updates the Q-values by applying (3.3). Instead of updating the Q-values iteratively with a matrix containing predefined rewards, we train the model in an online manner by getting the feedback from the network. This could have the disadvantage of a poor behavior at the beginning, but the results show that the tuning improves over the time. We also point out that we implemented fixed values for the rest of the parameters of the Q-learning algorithm during the experiment, that is $\gamma = 0.8$ and $\alpha = 0.5$.

Table 3.1. Buffer occupancy comparison

	Intelligent AQM		**Non-Intelligent AQM**	
	Average	***Maximum***	***Average***	***Maximum***
FQ-CoDel	1.60 %	2.70 %	2.09 %	2.80 %
CoDel	0.91 %	2.30 %	1.58 %	2.90 %

Another point to consider is the performance of our method in terms of the buffer occupancy at the router. Based on the statistics obtained from the Linux Traffic Control utility, we compare the percentage of buffer occupancy for each experiment in Table 3.1. Note that we take into account the set hard limit buffer size for the percentage calculation. In other words, the buffer occupancy would be 100% if the queue had 1000 packets at a specific instant. As can be seen, the buffer occupancy is lower when R1 employs our intelligent AQM, thanks to the balance between dropped/marked packets that the algorithm achieves over the time. Finally, we want to mention that the Python code of the experiments described in this subsection is publicly available at [84]. We intent to make our contribution accessible to researchers and developers who are actively working on congestion-related problems of the Internet. Please cite this work if you use any posted script for your own works.

3.4. Summary

As more end devices are getting connected, the Internet will become more congested. Various congestion control techniques have been developed either on transport or network layers. Active Queue Management (AQM) is a paradigm that aims to mitigate the congestion on the network layer through active buffer control to avoid overflow. However, finding the right parameters for an AQM scheme is challenging, due to the complexity and dynamics of the networks. On the other hand, the Explicit Congestion Notification (ECN) mechanism is a solution that makes visible incipient congestion on the network layer to the transport layer. In this work, we propose to exploit the ECN information to improve AQM algorithms by applying Machine Learning techniques. Our intelligent method uses an artificial neural network to predict congestion and an AQM parameter tuner based on reinforcement learning. The evaluation results show that our solution can enhance the performance of deployed AQM, using the existing TCP congestion control mechanisms.

Chapter 4

Federated Intelligence for Inter-Domain Congestion

4.1. Motivation

Communication over the Internet relies on data packet transmission across a selected network path, while involved over the complex interconnected network elements. To achieve this, different network elements of the Internet, *e.g.* routers, usually first place the received data packets in queues, where they wait their turn to be transmitted over the next determined link. When there are too many queued packets awaiting transmission, the buffers of the network element's interface may overflow and the involved link is said to be congested. Therefore, determining the proper buffer size is deemed as a key component to evade packet losses along network paths when congestion appears. While a large buffer could reduce packet losses, excessive buffering could lead to increased latency, as packets have to wait longer in the queues. This phenomenon is known as bufferbloat and causes poor performance at bottleneck links of today's Internet [60]. This effect can be tackled by the network elements through Active Queue Management (AQM) methods, which are designed to control the flow of the arriving packets and avoid network congestion. To achieve so, AQM schemes determine whether there is incipient congestion on the involved link and choose either dropping specific packets or marking them with "experienced congestion" labels. The main advantage of dropping packets with AQM rather than with tail-drop queues, *i.e.* non-AQM buffers, is to eliminate the unnecessary global synchronization of flows when a queue overflows. In this way, an AQM scheme can decide to drop packets when the network experiences incipient congestion in a controlled fashion. As a result, packets experience shorter delays, as their flows are regulated by the AQM mechanism in use, and the throughput is improved. Despite the advantages of AQM, it is not widely adopted on the network elements of the Internet Service Providers (ISPs), since the AQM mechanisms have parameters that might be difficult to tune in dynamic environments. Also, network elements with more memory available in the market have created the misconception that the larger the buffers, the better.

Accordingly, we proposed an intelligent method for implementing AQM in our previous work [85] by exploiting the standardized Explicit Congestion Notification (ECN): a process of making incipient congestion visible by exposing the presence of congestion on a path to network and transport layers through codepoints and flags in both IP and TCP headers. Our goal was to boost the alleviation performance that AQM techniques provide at bottlenecks by dynamically adjusting the AQM parameters and considering the specific network conditions. Therefore, we introduced a Machine Learning-based solution that comprises a Recurrent Neural Network to predict congestion and an AQM parameter tuner based on the Q-learning algorithm. The proposed scheme, however, was delimited to scenarios where only one router performs the intelligent AQM process (IAQM). For instance, a setting where an edge router predicts the congestion ahead, based on the ECN feedback that it receives from the core network, and then tunes its AQM parameters. As a result, the IAQM scheme dynamically reduces the Round-Trip Time (RTT) and increases the throughput of the connections being handled by the edge router.

In this work, we address the problem of congestion control by significantly enhancing existing AQM methods and taking into account the routers involved in inter-domain communications. This problem turns out to be even more challenging than a single-domain communication scenario, as each border router may not be able to receive ECN feedback in order to predict the congestion ahead. Additionally, a kind of cooperative mechanism is needed to achieve an effective Machine Learning solution where the privacy is paramount: an inter-domain link involves routers at several organizations or geographical regions, which means the possibility of having one or more domains not willing to share their data. That is why these domains are also known as Autonomous Systems (ASes), which consist of ISPs or Content Providers (CPs) communicating each other through an Internet Exchange Point (IXP), as depicted in Figure 4.1.

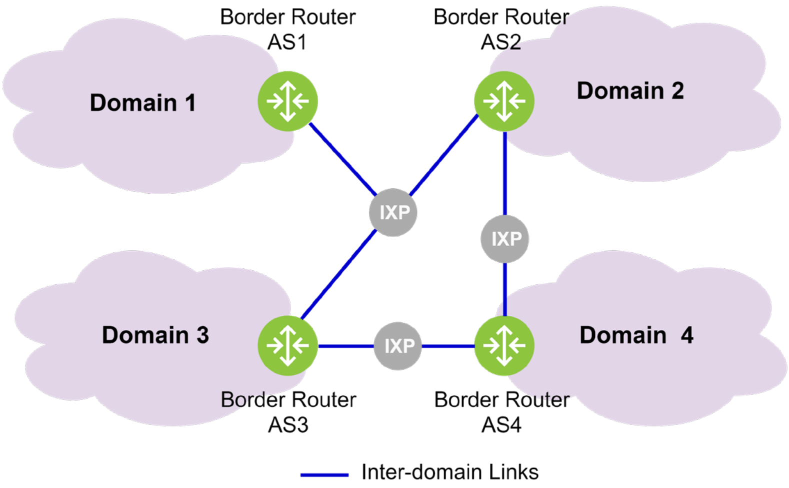


Figure 4.1. Example of an inter-domain communication scenario.

Managing congestion is an essential factor for an IXP and its connecting ASes. However, despite experiencing significant and persistent congestion at multiple peering links, both ASes and IXPs have no primary means of controlling congestion. That is, as the traffic sources and destinations are beyond its domain, a border router or an IXP cannot rely on the traditional congestion notification mechanisms such as ECN [86]. On the other hand, understanding the performance of the network elements requires measuring several parameters, such as utilization, loss rates, and variation in latency. Operators that control IXPs could measure such parameters for their links, although accurate assessment of these parameters may require cooperation of the operator at the other end of the links [87]. Moreover, the operators do not usually share this kind of information with their counterparts. For these reasons, we propose to apply the Federated Learning (FL) paradigm to intelligently address the inter-domain congestion problem.

FL is an approach where multiple entities collaborate in solving a Machine Learning problem, under the coordination of a central server or service provider [88]. To achieve the learning objective, each entity participates without exchanging private raw data, which are stored locally. The original emphasis of FL was on cross-device settings, *i.e.* mobile and edge devices applications [89], but FL has been applied to an increasing number of scenarios where a few and relatively reliable entities, such as the data centers of several organizations, collaborate to train a model [90], [91]. These kinds of scenarios are known as cross-silo settings. The main difference between the cross-device and cross-silo settings is that, in the former, a very large number of devices participate in the learning and their participation is likely to occur once in a task. On the other hand, in cross-silo settings only a small number of elements (typically, 2 to 100) contribute to the learning process by training a model on siloed data. In both cases, the data are generated locally and remain decentralized. At the same time, a central entity orchestrates the training process and receives the contributions of all entities. These characteristics make FL conceptually different from the decentralized and distributed learning approaches. A more detailed comparison of the FL settings versus the distributed and peer-to-peer learning can be found in [88]. It is also important to highlight that, different from many Machine Learning approaches, in FL the data are usually considered as unbalanced and not independently or identically distributed (non-i.i.d.) because each entity has different amount of local data to train on and these data rely on particular entities' behaviours [89]. Furthermore, depending on the distribution characteristics of the data, FL can be categorized as horizontal or vertical. In horizontal FL scenarios, the local datasets have the same feature space, but may have different sample ID space. In contrast, vertical FL refers to those cases where the datasets have the same sample ID space, but dissimilar feature space [92]–[94].

Accordingly, in this work we propose an intelligent scheme for AQM where the inter-domain congestion is predicted based on the horizontal FL approach. That is why we introduce our solution as the Federated Intelligence for AQM (FIAQM), whose key contributions are summarized as follows:

- A proof-of-concept study on non-static AQM. We demonstrate how the idea of dynamically tuning AQM parameters may boost the adoption of AQM mechanisms to mitigate the Internet's bufferbloat effect.
- An intelligent congestion control framework that is compatible with other solutions. Our proposed FIAQM leverages the benefits of using existing AQM mechanisms to control congestion over inter-domain links, which are not managed by a single party.

- A multi-domain learning approach in which local network data remains private. As in inter-domain scenarios privacy is a major concern, FIAQM allows the cooperation of two or more ASes to achieve common goals in terms of congestion by avoiding the sharing of raw data.
- A practical application of Deep Learning and FL in networking. We propose an adaptation of the FL algorithm that, along with a tailored neural network, effectively learns congestion levels of the link queues involved in cross-domain connections.
- An open-source environment for real-time evaluation. Finally, we evaluate the performance and feasibility of the FIAQM scheme in a setting that emulates a realistic inter-domain network communication, whose code is publicly available for further research and development.

Overall, a typical scenario for FIAQM comprises two border routers, which belong to different ASes, and an IXP. Each border router has intra-domain link buffers corresponding to the interfaces that connect them to other network elements within their own domains, as depicted in Figure 4.2. Both border routers exchange the aggregated parameters of the model to be trained with a central server, known as the Learning Orchestrator in our solution. We propose to place the Learning Orchestrator at the IXP premises, since it is supposed to be a neutral player. In this way, FIAQM applies FL to predict the IXP congestion based on the buffer statistics of the intra-domain links of the border routers involved (denominated as the Local Learners). The predicted IXP congestion is then used for the AQM parameter tuning of the inter-domain link buffers, similar to the tuning process introduced in [85].

The remainder of this chapter is organized as follows. We review the related work on inter-domain congestion in Section 4.2. In Section 4.3, we provide further details about the FIAQM architecture, whose evaluation performance results are discussed in Section 4.5. On the other hand, we explain the details of our experimentation design in Section 4.4.

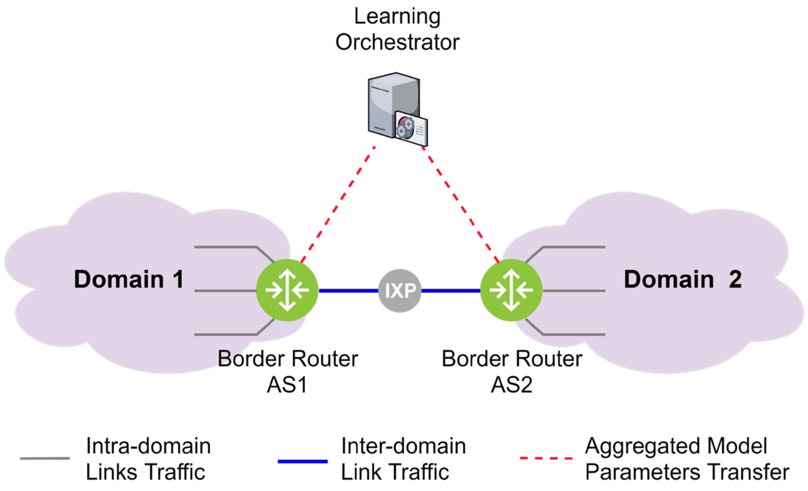


Figure 4.2. Typical scenario for the proposed FIAQM scheme.

4.2. Related Work

The inter-domain congestion control problem has been addressed from different perspectives. One common approach is to tackle the routing bottlenecks. These bottlenecks are inevitably caused by the Border Gateway Protocol (BGP), since the border routers tend to forward packets along the path with minimal routing cost. As a result, routing bottlenecks concentrate on a few links and happen to be asymmetrical, *i.e.* the inbound congestion does not correspond to the outbound one on the same link [95]. Therefore, the solutions for routing bottlenecks proposed in the literature mainly rely on dynamic load balancing, which can operate either on inter-domain or intra-domain links.

To this end, authors in [96] present a system to improve the ISPs network throughput by jointly optimizing intra-domain routes and inter-domain routes. Their solution provides an ISP and its neighbor CPs with a network abstraction on a virtual switch that allows to program requirements in a collaborative way. Conversely, an architecture for an efficient inbound traffic control based on the Software Defined Networking (SDN) paradigm is proposed in [97]. This architecture exploits the features of the OpenFlow protocol for network traffic engineering tasks in inter-domain routing. Similarly, Chiesa *et al.* describe the benefits of using the SDN approach for traffic engineering at IXPs. The authors explain how SDN enables such a network programmability that permits the members of an IXP to optimize their traffic load balancing and overcome the limitations of BGP [86]. Considering the privacy preservation in SDN-enabled scenarios for inter-domain traffic, authors in [98] propose a solution to avoid incorrect forwarding behaviours without exposing private routing information among domains. Likewise, [99] presents a mechanism for a dynamic end-to-end Quality of Service (QoS) coordination in multi-domain scenarios. This mechanism processes information in a distributed manner at the domain level and optimizes the routing by adaptively learning the results of past QoS requests.

It is important to highlight that a routing bottleneck is essentially different from a bandwidth bottleneck. The latter refers to the link with the smallest available bandwidth on a route, while the former is related to the number of routes carried by a link regardless the provisioned link capacity [100]. Even though they do not necessarily imply each other, routing bottlenecks can derive in bandwidth bottlenecks, which are the ones that ultimately cause the congestion that affects the networks' communication performance. For this reason, we address the inter-domain congestion problem with a focus on the bandwidth bottlenecks. This does not mean that our method cannot be used along with some of the described solutions for routing bottlenecks. Nevertheless, how to combine both approaches is beyond the scope of this work.

With regards to our learning setting based on buffer statistics, there is some literature about the use of queue measurements for congestion control improvement. For instance, authors in [101] propose a fine-grained queue measurement solution in the data plane for immediate control actions, which can support the deployment of new and more sophisticated AQM schemes. Using In-band Network Telemetry (INT) and traffic snapshots (fixed-sized time windows of traffic on a queue), their solution can determine the flows that consume large portions of a queue. Similarly, Li *et al.* propose a High Precision Congestion Control mechanism, which leverages the INT metadata reported by the routers along the path [102]. The metadata includes egress port metrics such as timestamp, queue length, transmitted bytes, and link bandwidth capacity to avoid congestion in high-speed networks. Although we acknowledge the value of the INT framework and its metadata, we consider not using INT in this work because it aims to monitor the performance of a core network within a single domain. However, we believe that the application of the INT metrics for the solution of an inter-domain problem, like the one presented in this chapter, could be a promising direction for a future work.

4.3. Architecture of FIAQM

In this section, we describe our solution in detail. Primarily, FIAQM consists of two principal modules: a congestion predictor and an AQM parameter tuner, like the IAQM solution presented in [85]. In FIAQM, however, the congestion ahead is predicted by means of the FL approach. This prediction is then utilized for the AQM parameter tuning of the inter-domain link buffers in both directions. Figure 4.3 depicts the overall architecture of FIAQM and the following subsections explain each component, respectively.

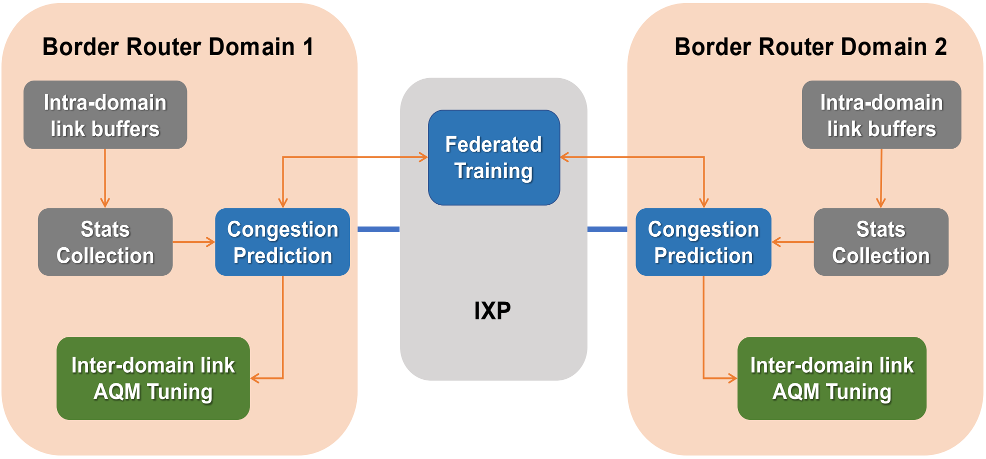


Figure 4.3. The FIAQM architecture for inter-domain congestion control. Main modules are replicated within each border router.

4.3.1. Federated Congestion Predictor

The first of the main components of the FIAQM architecture is a congestion predictor based on a Long Short-Term Memory (LSTM). An LSTM is a type of Recurrent Neural Network and deemed as an effective tool for time-series forecast. Its inputs include both the current sample and the previous observed sample, such that output at time step $t-1$ affects the output at time step t. Each neuron of the LSTM has a feedback loop that returns the current output as an input for the next step [74]. For these reasons, FIAQM employs an LSTM to predict congestion in a federated manner by considering drop rates at each queue per time interval as inputs. Hence, the drop rate x in a time interval i is calculated as follows:

$$x_{t_i} = \frac{D_{t_i}}{P_{t_i}} \qquad (4.1)$$

where D is the number or dropped packets and P the total packets arriving at the queue within each time interval. Additionally, we rearrange the vector of drop rates as an input matrix $\mathbf{X}$ corresponding to ten time steps and an output vector $\mathbf{y}$ of one time step, as shown in (3.1).

The rationale behind rearranging the samples in ten time steps is to improve the performance of the predictive model by having additional context. In this way, the estimation of drop rates contemplates more prior observations. Note that this data rearrangement is performed with the available samples of each queue participating in the FL training.

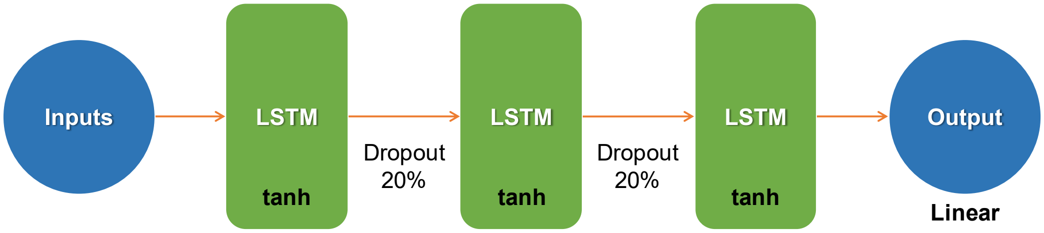


Figure 4.4. LSTM network structure for the FIAQM's congestion predictor.

The structure of the LSTM is similar to the one described in [85] and encompasses $L = 3$ hidden layers with 30 neurons each. The output layer employs a linear activation function while the hyperbolic tangent (tanh) is used as the non-linear activation function at the hidden layers, since it provides a three-state decision making (negative/neutral/positive) on what information to add or remove to/from the hidden cells [103]. Also, a dropout regularization of 20% is included at the output of each hidden layer, except the last one, in order to avoid model's overfitting, as shown in Figure 4.4. More specifically, each hidden layer $l \in [0, L)$ of the LSTM network computes the following function for each element in the input sequence [104]:

$$h_t^{(l)} = \tanh\left(W_{\mathrm{ih}}^{(l)} x_t^{(l)} + b_{\mathrm{ih}}^{(l)} + W_{\mathrm{hh}}^{(l)} h_{t-1}^{(l)} + b_{\mathrm{hh}}^{(l)}\right) \quad (4.2)$$

where $h_t^{(l)}$ is the hidden state at time t, $W_{ih}^{(l)}$ and $b_{ih}^{(l)}$ represent the weight and bias of the block input at layer l, and $W_{hh}^{(l)}$ and $b_{hh}^{(l)}$ are the weight and bias values of the hidden cells. Correspondingly, $h_{t-1}^{(l)}$ is the hidden state of the layer at time $t - 1$ and the input of the l-th layer, $x_t^{(l)}$, is the hidden state of the previous layer $h_t^{(l-1)}$ multiplied by the dropout of the previous layer, $\delta_t^{(l-1)} = 0.2$. Conversely, each output in the sequence is computed at the output layer through a linear function, as follows:

$$y_t = W_o\left(h_t^{(L-1)} + b_o\right) \quad (4.3)$$

where W_o and b_o are the weights and bias of the output layer, respectively, and $h_t^{(L-1)}$ is the state of the last hidden layer. The formulation of the LSTM presented above focuses on the activation functions for the hidden layers and the output layer to explain their relationship with the time steps. A more detailed formulation regarding the rest of the components of the LSTM architecture can be found in [105].

The Learning Orchestrator performs the global training of the LSTM model, which is used for the congestion prediction of the inter-domain link in each direction. In this way, the proposed LSTM-aided Federated Congestion Predictor (FCP) functions as follows:

each router has a fixed local dataset that differs from the other router's dataset, since they might have different number of intra-domain links with dissimilar levels of queue drop rates. At the beginning of each learning round, the Learning Orchestrator sends the current global model state to the routers, also known as the Local Learners in our solution. Next, each router performs a local computation based on the global state and its local dataset and, afterwards, sends an update to the orchestrator. Finally, the Learning Orchestrator applies the updates received from the Local Learners to its global state and the learning process repeats.

Due to the nature of our problem, we employ a cross-silo FL since individual routers or group of routers might belong to different proprietary networks. Our learning model is intended to be trained across these silos without exchanging raw data, which may represent ASes private information or a single organization's data that cannot be centralized between different geographical regions. Additionally, we consider the routers data as unbalanced and non-i.i.d., as well as the synchronous model updates that proceed in rounds of communication, as presented in [89]. The canonical FL problem involves learning a single, global statistical model from data stored on remote entities. For our problem, we aim to learn this model under the constraint that border routers data are stored and processed locally, with only intermediate updates being periodically communicated to the Learning Orchestrator. In particular, the goal is to minimize the objective function for the global learning [94], as follows:

$$\min_{w} F(w) \coloneqq \sum_{k=1}^{M} p_k F_k(w) \tag{ 4.4}$$

where w represents the model parameters, *i.e.* the weight and bias values of the hidden and output layers of the LSTM network. In our scenario, M is total number of queues involved in the congestion prediction process and p_k is the relative impact of each queue. On the other hand, F_k is the local objective function for the learning on the k queue, as follows:

$$F_k = \frac{1}{n_k} \sum_{j_k=1}^{n_k} f_{j_k}(w; x_{j_k}, y_{j_k}) \tag{4.5}$$

where n_k is the number of samples available locally.

Algorithm 4.1. Federated Congestion Predictor (FCP)

```
$q \leftarrow$ set of queues with non-zero drop rate data
for each round $r = 1, 2, 3, \dots, \Gamma$ do
  $u \leftarrow$ random subset, $u \in q$
  for each queue $k = 1, 2, \dots, M \in u$ in parallel do
    get $w$ from Learning Orchestrator
    $w_k \leftarrow w$
    $d_k \leftarrow$ count $n_k\ \forall\ x_{j_k} \neq 0$
    for each local training iteration $z = 1, 2, 3, \dots, Z$ do
      $w_k \leftarrow w_k - \eta \nabla F_k$
    return $w_k$ and $d_k$ to Learning Orchestrator
  $p_k \leftarrow d_k / \sum_{k=1}^{M} d_k,\ \forall\ k$
  $w_{t+1} \leftarrow \sum_{k=1}^{M} p_k w_k$
```

To solve this federated optimization problem, we adapt the Federated Averaging (FedAvg) algorithm presented in [89]. Accordingly, the algorithm combines a local stochastic gradient descent computed with the data of each queue at each border router and a model averaging performed by the Learning Orchestrator. The adaptation of the FedAvg algorithm for our proposed FCP is detailed in Algorithm 4.1, where η is the learning rate, which is assumed to be the same for all the Local Learners. It is important to highlight that d_k contains the number of data samples with non-zero values. The rationale behind this idea is that queues with higher drop rates affect the parameter averaging with higher values of relative impact p_k. In this way, the federated LSTM model learns more from those queues with non-zero drop rates for the congestion prediction. On the contrary, the queues with a few or zero samples of congestion data make a little or no contribution to the learning process.

4.3.2. AQM Parameter Tuner

In general, the parameters of the AQM algorithms are set to values that yield a reasonable performance for the typical network conditions. However, AQM mechanisms are expected to allow parameters adjustment depending on the specific characteristics of a network and their interactions with other network tasks over time [77]. Consequently, we embrace the idea of adjusting the AQM parameters according to the network's changing circumstances, so that the performance is dynamically improved, as well. Nevertheless, the achievement of this goal can end up in a very complex job and that is the main reason why network managers prefer not to use AQM at all. Another point to consider is the right metric to evaluate the effectiveness of a resource allocation/configuration in a network. The key metrics to be considered for queue management are, usually, throughput and delay. Accordingly, the objective is to minimize the delay and maximize the throughput. It turns out that, trying to increase the throughput by allowing as many packets into the links as possible, results in a rising length of the queues and, therefore, longer delays. As an alternative, a separate metric that combines throughput and delay can be taken into account. That is why the ratio of throughput, T_{put}, to measured RTT, m_{RTT}, has been proposed by network designers as a metric to evaluate the effectiveness of a resource configuration, such as the AQM parameters. This throughput-to-delay ratio is also known as the power of the connection, $P_c = T_{put}/m_{\text{RTT}}$, and, even though this metric has some limitations, it is widely accepted for evaluating the network resource configuration effectiveness [106], especially the queue management for congestion control [107]. Maximizing P_c is, however, a non-trivial task considering the network dynamics.

For the reasons explained above, we model the AQM parameter-tuning problem as a Markov Decision Process (MDP). In the FIAQM scheme, the decision process is based on the inferred congestion ahead, *i.e.* the output of the FCP described in Section 4.3.1. In this way, we define the states S as a set of discrete levels of congestion that the inter-domain link will be likely to experience, the set of actions A comprises specific values of the target parameter of the AQM algorithm in use, and the reward R depends on P_c. In our scenario, each border router acts as the agent that makes the decisions. This way, our method can adjust the target parameter so that more packets are dropped proactively and in a controlled manner at the sending border router, as they will be likely dropped ahead in the other domain. In other words, the AQM parameter tuner is modelled as an MDP with the objective of finding an optimal behavior that maximizes P_c. To do so, we utilize the Q-learning algorithm [79], which defines the function $Q(S, A)$, representing the quality of a certain action in a given state, and that is defined by (3.3)

This equation characterizes the maximum future reward of present state s and action a in terms of immediate reward and maximum future reward for the next state $\bar{s}$ and action $\bar{a}$. In this manner, the Q-learning algorithm iteratively approximates the function $Q(S, A)$, as shown in Algorithm 4.2. More specifically, our AQM parameter tuner observes current and next states as levels of congestion, *i.e.* the predicted drop rates of the link buffer at the router in the destination domain. Additionally, both current and next states are discretized to delimit the complexity of the environment. Finally, the actions are a set of predefined values for the target parameter of the specific AQM in use. As the agent does not know what action to take at the beginning, there is an initial stage of exploration, which depends on the parameter ε. The value of this parameter determines if the Q-learning algorithm prefers to explore random actions rather than exploit the historical data to take an action.

Algorithm 4.2. AQM Tuner

```
$S$ ← set of discretized values of predicted congestion
$A$ ← set of AQM target parameter values
$Q(S, A)$ ← Q-table initialization
$\varepsilon$ ← exploration/exploitation rate, $\varepsilon \in [0,1]$
$s$ ← get state from FCP, $s \in S$
for each period $T = 1, 2, 3, \ldots$ do
    if random number < $\varepsilon$
        then $a$ ← select a random action, $a \in A$
        else $a$ ← $\mathrm{argmax}_a Q(s, A)$
    change parameters according to $a$
    $m_{\mathrm{RTT}}$ ← measure delay
    $T_{put}$ ← measure throughput
    $R$ ← $P_c$
    $\bar{s}$ ← get state from FCP, $\bar{s} \in S$
    update $Q(S, A)$
    $s$ ← $\bar{s}$
```

4.4. Experimentation Design

In order to evaluate our FIAQM scheme, we set up a network emulation environment on Mininet to run experiments and obtain more realistic results. We chose Mininet as the tool to validate our prototype since it allows a flexible SDN environment with high degree of confidence for real-time tests [108]. Moreover, Mininet eases the sharing of our solution, which could be deployed into a real production network using our code and test scripts, publicly available at [109]. Accordingly, our emulation network consists of two border routers and 20 hosts connected to each one, forming a dumbbell topology. In this way, there are 20 pairs of hosts generating traffic from one domain to the other (hosts of each pair are in different domains). Figure 4.5 depicts the implementation of our experimentation setting. Note that for simplicity, only one direction of the learning process for the congestion prediction is depicted, that is, considering traffic from Domain 2 to Domain 1. Therefore, the IAQM tuning happens at the egress buffer of the Border Router Domain 2 in this setting.

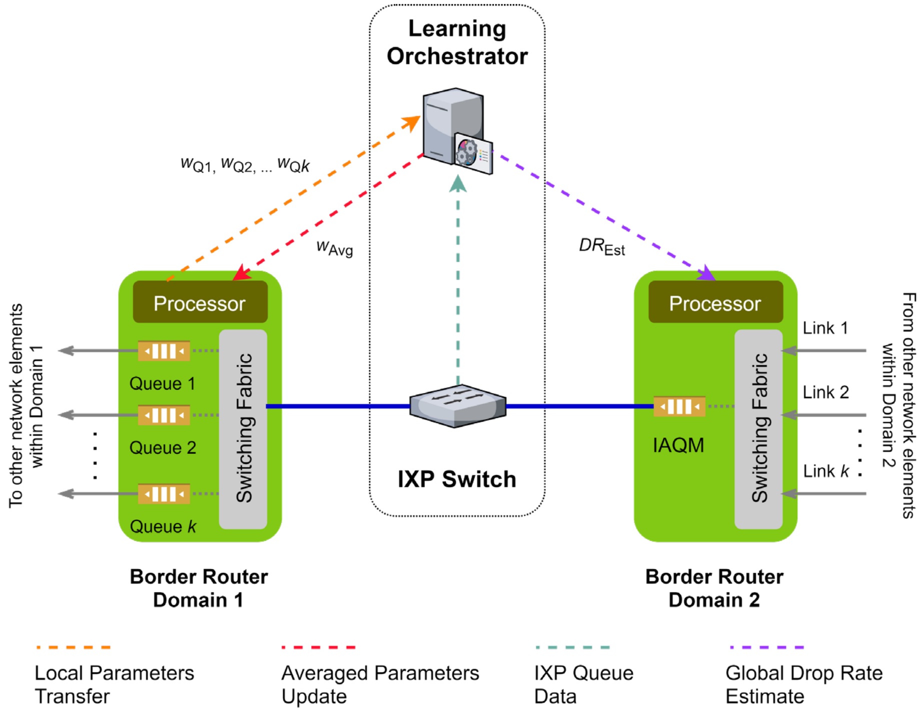


Figure 4.5. Implementation of the FIAQM for experimentation.

With respect to the FCP implementation, our environment involves three Mininet hosts acting as the Learning Orchestrator and two Local Learners, the latter being represented by the processor block at each border router. Additionally, PyTorch is employed on these hosts for the execution of the learning process as described in Algorithm 4.2. We chose PyTorch as the framework for the implementation of our FCP algorithm because it provides a high level of control and flexibility, which we weigh as a key feature for our network emulation. Moreover, PyTorch's usability and developer-centric design facilitates the implementation of new Deep Learning architectures, using the familiar concepts developed for general purpose programming languages such as Python [110]. This is particularly relevant for the application of the FL approach, since it needs to be deployed in a distributed manner when implementing real-world setups. We see this fact as a significant advantage of PyTorch over other Deep Learning frameworks like TensorFlow. For example, we employed TensorFlow Federated (TFF) for the fulfilment of the FL version of the LSTM model proposed in [85]. We were able to confirm that TFF only enables the simulation of FL models with decentralized datasets, as stated in [111], but not an actual distributed deployment. For these reasons, we decided to utilize PyTorch as the Deep Learning framework for the validation of our FIAQM scheme.

In relation to the traffic generation between the host pairs for the queue metrics, we use the NetPerf tool [112], which allows to stress the network under a combination of several types of IP traffic. Furthermore, we perform tests according to the Real-time Response Under Load (RRUL) Specification to emulate a more core-network-like IP traffic. In fact, RRUL-based tests reliably saturate the measured link and, therefore, exposes any presence of the bufferbloat effect. To this end, the RRUL specification contemplates simultaneous bidirectional TCP and UDP streams, VoIP-like streams, multiple up/down TCP streams to shorten the ramp-up-to-saturation period, running traffic long enough to defeat bursty bandwidth optimizations, and test server(s) within 80 ms of testing client(s) [113]. Next, the emulator collects the buffer statistics in intervals of 100 ms using the Linux Traffic Control (TC), since this utility lets monitor the queue events generated by the kernel [114]. The value of the time interval corresponds to the typical assumption for a single RTT interval in IP networks.

Subsequently, we use set both the intra-domain and the inter-domain link buffers to a relatively small hard limit of 1000 packets. This assumption is based on the fact that small buffer sizes in backbone routers are sufficient for many networks and recommended for overall scalability [80], [81]. Additionally, all the intra-domain link buffers are configured with AQM. More specifically, we consider the Flow Queue - Controlling Queue Delay (FQ-CoDel) whose target parameter to configure is the acceptable minimum standing/persistent queue delay [34]. As this parameter decreases, more packets are dropped in a controlled manner, since they are supposed to stay for shorter times in the queue. Consequently, there are less packets in the queue and the link delay decreases. On the other hand, when the FQ-CoDel target parameter is high, the scheme does not drop packets and there is a higher delay due to longer queues. Also, packets start to be dropped uncontrollably as the queue overflows and, therefore, the throughput is deteriorated.

As a preliminary experiment, we show that the drop rate data of the queues at the Border Router Domain 1 describe dissimilar patterns, as depicted in Figure 4.6a. Therefore, the traffic data generated by the RRUL test and gathered with the TC utility exhibits the kind of non-i.i.d. behaviour necessary for the FL model of the FCP. For the sake of clarity, we depict the drop rate data corresponding to ten queues only, but similar graphs are obtained when more queues are considered. On the other hand, to show the influence of tuning FQ-CoDel, we set up a simple test that consists of modifying its target and interval parameters at the egress buffer of the Border Router Domain 2 while data are constantly transferred between two hosts, each one in a different domain. The interval parameter ensures that the measured minimum delay does not become too old and, typically, the target delay is 5% of this interval [64]. Therefore, we set FQ-CoDel with target values from 1 ms to 6 ms and intervals from 20 ms to 120 ms, respectively. As can be seen in Figure 4.6b, although an AQM scheme such as FQ-CoDel is meant to operate unchangeably, there is a noticeable effect when its target parameter varies: both m_{RTT} and T_{put} are affected by the target delay configuration. This is consistent with our solution formulation explained in Section 4.3.2.

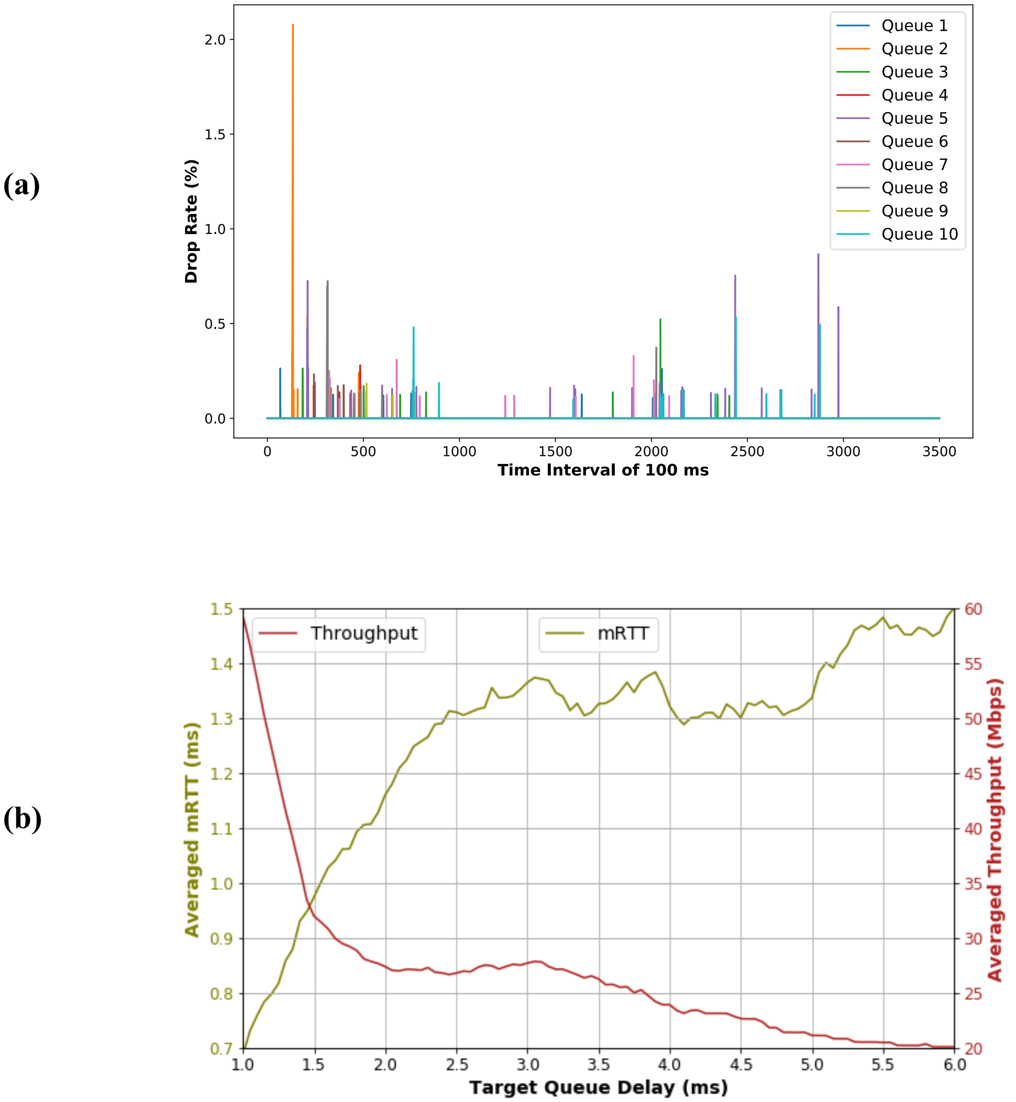


Figure 4.6. Preliminary tests for the Experimentation Design. a) Queues data at Border Router Domain 1. b) Effects of tuning FQ-CoDel target parameter on $\boldsymbol{m}_{\mathbf{RTT}}$ and $\boldsymbol{T}_{\boldsymbol{put}}$.

For the parameters exchange between the Learning Orchestrator and the Local Learners, we use the Secure File Transfer Protocol (SFTP), which runs over the Secure Shell (SSH) protocol, to avoid sending the parameters in the clear. SFTP protects the data integrity through cryptographic hash functions and provides authentication for both the server and the client [115]. In this way, we also consider security concerns in a real inter-domain scenario by adding encryption functionality to the communication between the parties involved in the FCP. Additionally, we assume that the pair of private and public keys have been shared prior to the execution of the Algorithm 4.1 and that a different port from the default SSH port, *i.e.* port 22, is agreed for the transfer.

Table 4.1. Model parameters to be transferred for the FCP.

Parameter	Description	Tensor Dimension
$W_{\mathrm{ih}}^{(0)}$	Weights of block input, hidden layer 0	120×1
$b_{\mathrm{ih}}^{(0)}$	Bias of block input, hidden layer 0	120
$W_{\mathrm{hh}}^{(0)}$	Weights of hidden cells, hidden layer 0	120×30
$b_{\mathrm{hh}}^{(0)}$	Bias of hidden cells, hidden layer 0	120
$W_{\mathrm{ih}}^{(1)}$	Weights of block input, hidden layer 1	120×30
$b_{\mathrm{ih}}^{(1)}$	Bias of block input, hidden layer 1	120
$W_{\mathrm{hh}}^{(1)}$	Weights of hidden cells, hidden layer 1	120×30
$b_{\mathrm{hh}}^{(1)}$	Bias of hidden cells, hidden layer 1	120
$W_{\mathrm{ih}}^{(2)}$	Weights of block input, hidden layer 2	120×30
$b_{\mathrm{ih}}^{(2)}$	Bias of block input, hidden layer 2	120
$W_{\mathrm{hh}}^{(2)}$	Weights of hidden cells, hidden layer 2	120×30
$b_{\mathrm{hh}}^{(2)}$	Bias of hidden cells, hidden layer 2	120
W_{o}	Weights of output layer	1×30
b_{o}	Bias of output layer	1

The use of SFTP is sufficient for the needs of our experimentation, since w_k are transferred as a Python dictionary with the parameters of the FCP model. The size of this dictionary is 77.8 kB and it contains the PyTorch tensors with the weight and bias values, whose dimensions are specified in Table 4.1. On the other hand, d_k is a Python list with u elements. For private and secure transfer of high-dimensional parameter vectors in a FL setting, which is not the case of this work, we point the reader to other research papers such as [116], [117]. It is also important to highlight that, although the FIAQM is tested in a distributed setting, the FCP algorithm is synchronously executed between the Learning Orchestrator and the Local Learners. This means that our experimentation design considers the coordination of the learning algorithm execution along with the transfer of the parameter files.

4.5. FIAQM performance evaluation

To evaluate our FIAQM scheme, we first demonstrate how the FCP algorithm predicts congestion accurately as a stand-alone entity. Next, we illustrate how the FCP integrates with the AQM parameter tuner to attain the objective of reducing congestion and improving the performance of an inter-domain connection.

4.5.1. FCP algorithm predictions accuracy

The experiments of this subsection are conducted in an offline setting with data previously gathered during the preliminary tests described in Section 4.4. Hence, we count on 21 datasets: one from the IXP queue, corresponding to the link between the IXP switch and Border Router Doman 1, and 20 from the queues of the intra-domain links of the aforementioned router. Subsequently, we train the FPC with η=0.001 and u=2, which means that two queues of the router are randomly selected to average the model parameters in each round. Also, the number of training rounds are set to Γ=10 and the local iterations to Z=1000. To make predictions, we utilize the data from the IXP queue as the set of test samples. Figure 4.7 shows how the predicted congestion of the FPC model, trained with the queue data of Border Router Domain 1, resembles the actual congestion of the IXP's queue. Note that, rather than predicting the exact value of drop rate in a particular time interval, we are more interested in capturing the tendency of that value. Hence, the predictions are accurate enough for our goal. In terms of the loss metric, we chose the Mean Square Error (MSE), which yields a value of 0.002 over the test subset.

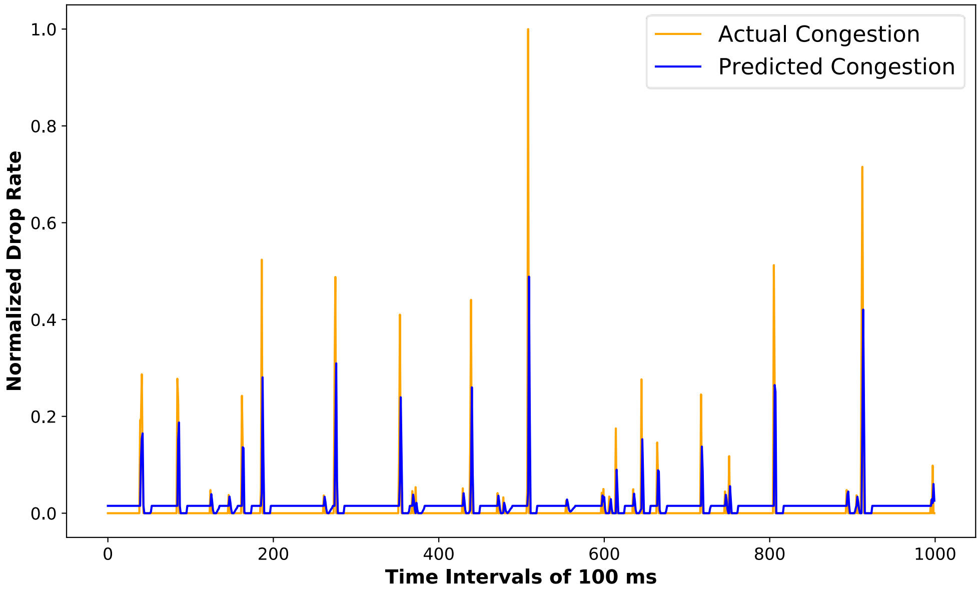


Figure 4.7. Actual congestion of the IXP queue and predicted congestion by the FCP.

On the other hand, we compare the loss obtained when the congestion predictor is trained in a federated fashion and in a centralized manner. As this comparison requires more exhaustive tests, we change the emulation parameters Γ and Z to 50 and 2000, respectively. We also run a separate centralized model that is trained with data from the IXP's queue. As can be seen in Figure 4.8, FCP gets lower cumulative loss than the LSTM model of the centralized congestion predictor. What is interesting about this result is that both federated and centralized models are evaluated by making predictions over a test subset from the IXP's queue. That is, the FCP outperforms the centralized congestion predictor, even though the test data is a subset of the dataset used for the centralized model training. This result is consistent with those presented in [89].

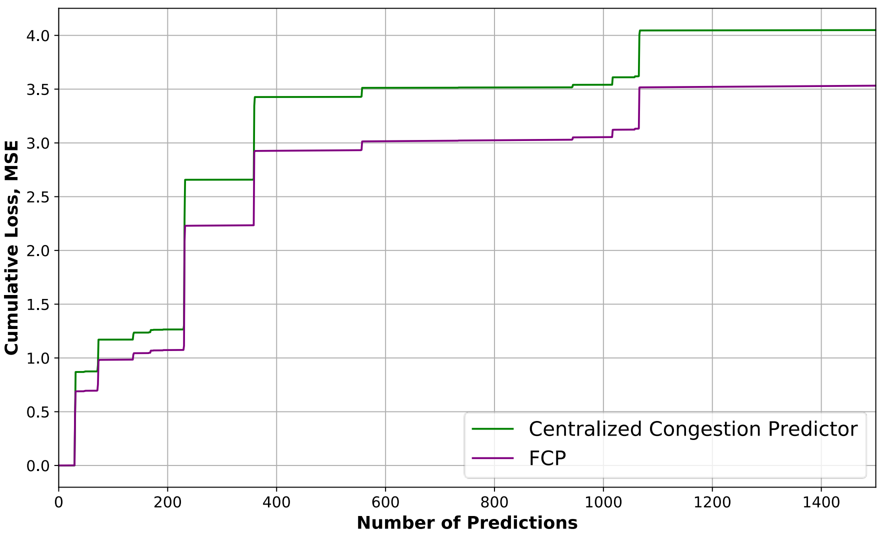


Figure 4.8. Evaluation loss comparison between a centralized congestion predictor and the FCP algorithm.

In contrast, the time complexity of the FedAvg algorithm can be expressed in terms of the total training rounds, Γ, local epochs, E, and the number of local samples, n_k, as $\mathcal{O}(\Gamma \times E \times \max_k(n_k))$. This means that the time taken for the FL training depends on the slowest participant in each round, also known as stragglers, because of the number of local updates that those participants need to execute [118]. In our proposed FCP algorithm, we reduce this complexity by considering that all the participants have the same number of local samples, that is $N = n_k$. It is important to highlight that this is a realistic consideration, since the traffic in core networks is very high and the routers' queues are likely to expose congestion frequently. In this way, the local epochs and local batches of the FedAvg algorithm are converted into Z local training iterations in FCP (step 8, Algorithm 4.1), which correspond to n_k. In other words, different from the FedAvg algorithm, in the FCP algorithm every participant happens to have the same number of local updates (or local training iterations, Z), which yields a time complexity of $\mathcal{O}(\Gamma \times N)$. Nevertheless, we show that $\Gamma \ll N$ is generally the case for our problem scenario.

To this end, we set various target loss values in order to determine how many rounds of training the FCP needs to reach those targets. Thus, four benchmarks are defined based on the cumulative loss over 2000 predictions as targets. In this experiment, the number of local iterations is $Z = 2000$, as well. Similar to the evaluation test explained previously, the predictions are made considering a test subset from the IXP's queue. We then compare the number of training rounds needed by the FPC algorithm against an LSTM trained in a centralized host, Figure 4.9. It is important to point out that, for the sake of the comparison, the term training rounds means the equivalent of training iterations for the centralized predictor. As this predictor acts alone, there is no real rounds of training.

As can be seen, the FCP algorithm requires less rounds during the training process to attain the desired loss on the test data. This result shows that, although there is an overhead in the congestion predictor training of the FIAQM, the proposed algorithm compensates this overhead by enabling a lighter training process in terms of the rounds needed. Moreover, this outcome evidences that the complexity of the FCP algorithm is heavily influenced by the number of samples used for the training process, N, rather than Γ.

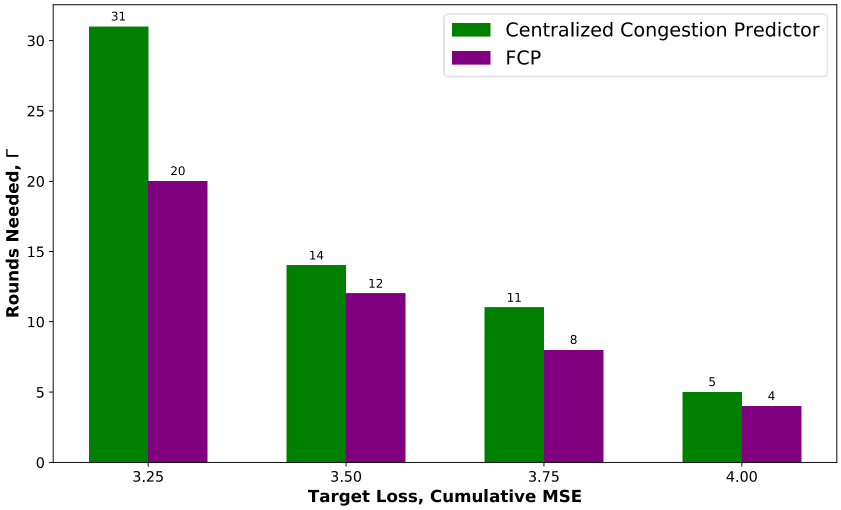


Figure 4.9. Number of training rounds needed to reach the target loss by a centralized congestion predictor and the FCP algorithm.

4.5.2. Real-time AQM tuning with FIAQM

In this subsection, we elaborate about the experiments that we conducted in real time to show the performance of our proposed method as a whole, that is, the FIAQM's main components working together. To this end, we carry out several experiments in the emulation setting described in Section 4.4. The network emulation parameters for this evaluation are summarized in Table 4.2. We assess the MDP for the AQM tuning problem by considering 100 levels of congestion as current or next states. To determine their levels, we keep the maximum observed and predicted values as reference for the discretization. We also delimit the actions to 100 values, which are the target delay of FQ-CoDel. In this way, the possible actions to take are a set of values from 1.1 ms to 11 ms in steps of 100 μs. As we explained in Section 4.3.2, we modify two parameters at the same time: the target delay and the interval. Thus, the experiments are more consistent as these two parameters are tightly related. For this assessment, the Border Router Domain 2 performs the IAQM while the Border Router Domain 1 is configured with the default target and the interval parameters in the Linux kernel: 5 ms and 100 ms, respectively.

Table 4.2. Emulation parameters for the evaluation of the FIAQM scheme in real time

Network Emulation Parameter	Value
Border Router Domain 1 - IXP link bandwidth	1 Gbps
Border Router Domain 2 - IXP link bandwidth	1 Gbps
Intra-domain links bandwidth	Random integer, [250, 500) Mbps
Border Router Domain 1 - IXP link delay	2 ms
Border Router Domain 2 - IXP link delay	2 ms
Intra-domain links delay	Random integer, [2, 10) ms
Number of hosts per domain	20
Buffers hard limit (all queues)	1000 packets

AQM mechanism (all queues)	FQ-CoDel
Intra-domain links AQM target (static)	2 ms
Intra-domain links AQM interval (static)	40 ms
Period of AQM parameters tuning, T	2 s
Emulation time	600 s

In terms of the FIAQM execution, the FCP runs in the background while the AQM tuner performs its job in an online manner. To achieve so, the Q-values are updated iteratively every 2 seconds based on both the predicted level of congestion ahead and P_c, which is calculated from the T_{put} and m_{RTT} values that two monitoring hosts, one in each domain, measure with active probes. Once the reward based on P_c is known, the algorithm updates the Q-values by applying (3.3).

On the other hand, the FCP utilizes pre-trained model parameters while the first training round is completed. Thus, the FCP predictions during this time are accurate enough for the AQM tuner. Additionally, 100 samples of the IXP's queue data are considered for the predictions, which means the historical levels of congestion in the past 10 seconds. Those predictions are transferred from the Learning Orchestrator to the Local Learner asynchronously, in form of a NumPy array of dimension 100 × 1 and 928 B in size. This array corresponds to the global drop rate estimate of the other domain, $\mathrm{DR}_{\mathrm{Est}}$, as depicted in Figure 4.5. In this way, the AQM tuner takes into account the most recent available values of $\mathrm{DR}_{\mathrm{Est}}$, even if the FCP is still processing a new training round.

Accordingly, Figure 4.10 shows the results of the real-time network emulation in 600 s. Note that, for the comparison sake, we set the FIAQM's tuner to start operating at 150 s of the emulation. Then, the AQM parameters of Border Router Domain 2 are fixed to the default values during the first 150 s and, from this time on, the IAQM tunes these parameters according to Algorithm 4.2.

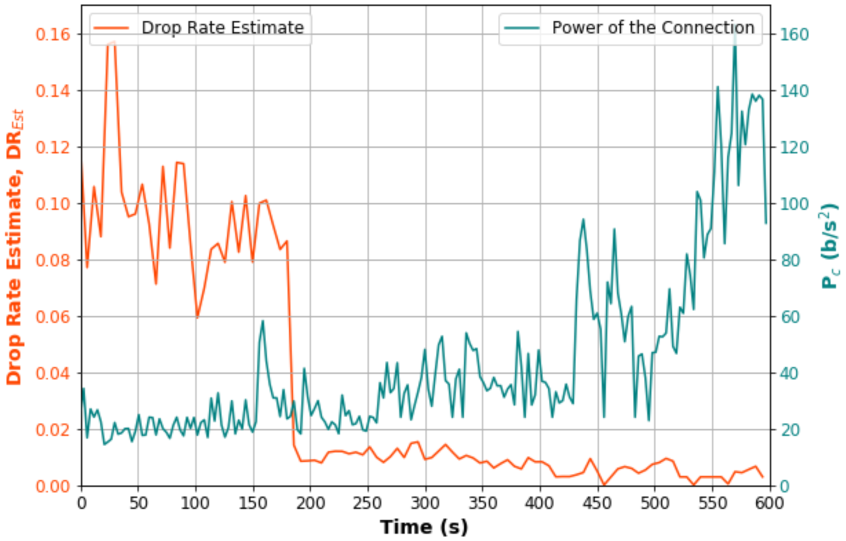


Figure 4.10. Improvement over time provided by FIAQM in terms of congestion reduction and $\boldsymbol{P_c}$ growth. AQM tuning starts at 150 s.

As can be seen, the drop rate ahead at the Border Router Domain 1, which corresponds to the $\mathrm{DR}_{\mathrm{Est}}$ values forecasted by the FCP, decreases significantly once the FIAQM starts the tuning process. Conversely, P_c tends to get higher values as the AQM tuner improves over time. As a result, the tuning process populates the Q-table with the values of P_c in the respective (s, a) coordinates at every iteration of Algorithm 4.2. We highlight that, thanks to the way that we design the AQM tuner, the resulting Q-table is a light NumPy array of 100 × 100 elements and 39.1 kB in size.

Finally, Table 4.3 summarizes the hyperparameters of both modules of the FIAQM scheme utilized for its evaluation in the real-time emulation. It is also important to point out that, although we designed our experimentation setting to make it as realistic as possible, Mininet has some limitations regarding the links bandwidth of the emulated network elements. In actual backbone networks, link data rates are of the order of tens or hundreds of Gbps. However, Mininet emulations are constrained by the data rate of the computer's network interface where Mininet is running and the number of emulated network interfaces. This means that, in order to achieve results that resemble real-world networks, this data rate capacity must be considered for all the links in the emulation environment. Nevertheless, the emulation parameters can be easily scaled when running our setting on other computers, actual SDNs, or even Linux-based bare metal routers [119]. Last but not least, we would like to remind the reader that the code of the experiments described in this subsection is publicly available at [109]. We intent to make our contribution accessible to researchers and developers who are actively working on congestion-related problems of the Internet. Please cite this work if you use any posted script for your own works.

Table 4.3. Hyperparameters of the FIAQM'S learning modules.

Module	Hyperparameter	Value
FCP	LSTM hidden layers, L	3
	Cells per LSTM hidden layer	30
	LSTM dropout regularization, δ	0.2
	Learning rate, η	0.001
	Subsets of non-zero queues, u	2
	Local training iterations, Z	1000
	Training rounds (maximum), Γ	10
AQM Tuner	Learning rate, α	0.5
	Discount factor, γ	0.8
	Exploration/exploitation rate, ε	0.5

4.6. Summary

Active Queue Management (AQM) has been considered as a paradigm for the complicated network management task of mitigating congestion by controlling buffer of network link queues. However, finding the right parameters for an AQM scheme is very challenging due to the dynamics of the IP networks. In addition, this problem becomes even more complex in inter-domain scenarios where several organizations interconnect each other with the limitation of not sharing raw and private data. As a result, existing AQM schemes have not been widely employed despite their advantages. Therefore, we present a solution that tackles the challenges of tuning the AQM parameters for inter-domain congestion control scenarios where the network management goes beyond an organization's domain. We then introduce the Federated Intelligence for AQM (FIAQM) architecture, which enhances the existing AQM schemes by leveraging the Federated Learning approach. The proposed FIAQM framework is capable of dynamically adjusting the AQM parameters in a multi-domain setting, which is hard to achieve with the conventional AQM solutions working alone. To this end, FIAQM uses an artificial neural network, trained in a federated manner, to predict beyond-own-domain congestion and an intelligent AQM parameter tuner. The evaluation results show that FIAQM can effectively improve the performance of the inter-domain connections by reducing the congestion on their links while preserving the network data private within each participating domain.

Chapter 5

Efficient Network Telemetry based on Traffic Awareness

5.1. Motivation

With the advancement of Software-Defined Networks (SDN) paradigm and the development of its programmable data plane (PDP) technologies, the network telemetry (NT) notion has emerged differing from the traditional network measurement schemes, as it comprises an automated process for remotely gathering and processing network data [120]. Moreover, traditional network monitoring technologies usually rely on active probes that are protocol-specific, such as Internet Control Message Protocol (ICMP) and Simple Network Management Protocol (SNMP) packets, or passive methods of measurements, which are based only on observations of undisturbed and unmodified packet streams of interest [121]. That is why NT is then deemed as a suitable answer to the challenges that the traditional network measurement technologies face in terms of adequate network visibility with better scalability, accuracy, and coverage, as well as hardware and protocol independencies.

The study of how to get high-quality network measurement data at low cost is important, since NT produces massive data in real network environments. The main goal of any NT scheme is to generate and collect measurement data locally at network nodes, depending on different service requirements, and transmit those data to a centralized controller for enabling an optimal network management. Therefore, an efficient telemetry deployment strategy is needed to compensate for the network performance loss due to the impact of gathering and transmitting the telemetry data itself. Networks' failures and performance problems can have a variety of causes, which requires different types of information to diagnose. That is why the ideal telemetry scenario contemplates the gathering of all the fine-grained data at a fine time scale. However, this means a high cost in terms of communication overhead. On the other hand, network managers need to get the telemetry information in a timely manner to quickly identify, isolate, and fix performance problems in order to minimize the impact on users and organization's revenue. Yet, it is difficult to measure many flows and packets with constrained resources at the network elements, which focus more on control functions such as packet forwarding. Since NT not only processes all the packets but also stores information about the packets, NT sometimes requires even more resources than the control functions do.

Today's NT practices follow a bottom-up approach, *i.e.* network managers collect data from network elements, aggregate it in a centralized collector, and extract the information they need. This approach poses several problems like having too many data to process. For this reason, a new approach is needed, one that provides network managers with abstractions of the metrics they are interested in [122]. Based on those interests, the granularity of the measurements should be different allowing to minimize the overhead produced by the telemetry data's transmission. In this way, different levels of measurement accuracy can also be obtained considering the network resources' limitations. Nevertheless, the task of matching network managers' desires with specific telemetry granularities might be challenging due to the network's changing conditions.

Moreover, NT applications only care about the telemetry data, instead of how to obtain those data. Then, a sort of telemetry tasks orchestration should be used in order to achieve efficient tasks distribution and telemetry data acquisition. In addition to upper-level monitoring applications, the orchestration of NT tasks should consider real-time and changing network flows. Nevertheless, how to achieve high-quality network measurement at low cost according to the existing network status is a key issue of NT that needs further research and development [120].

We then propose to address the problem of efficiently gathering NT data through a modular framework that is independent of the NT scheme in use. The core of our solution is Machine Learning-based NT Controller, which autonomously decides the granularity of the measurements to be transmitted. This decision is made taking into account network managers' needs and the traffic that a network element is experiencing. To achieve so, we consider an anomaly detection mechanism, which aims to discover unexpected events in the traffic data. In this way, several types of traffic are identified and the telemetry data are selectively transmitted based on those traffic types.

Accordingly, our proposed mechanism utilizes a classifier to detect anomalous behaviours in the traffic that a network element is forwarding. The classification model considers the traffic characteristics that common cyberattacks expose, so that the flows are segmented in different types (including benevolent traffic) based on those characteristics. Thus, our design aims to classify the network traffic anomalies and, according to this segmentation, decides the level of granularity of the telemetry data that a network element should transmit. Our rationale behind this proposal is that malicious traffic patterns can be exploited to determine the frequency in which NT data should be sent. In other words, when normal patterns of flows are detected, there is no need for a very fine granularity in the NT data gathering. In this way, for example, the queue occupancy measurements are not to be transmitted very frequently unless malicious traffic is negatively affecting the network elements' buffers. In fact, this kind of approach has been researched in the literature. For instance, authors in [123] study the behaviour of some network performance metrics, such as the buffer occupancy, as a consequence of malevolent traffic produced by attacks like Denial of Service (DoS), Distributed Denial of Service (DDoS), and SYN/TCP flooding (a type of DoS/DDoS flood attack using the TCP protocol). Therefore, we aim to take advantage of such a relationship between the traffic patterns that typical cyberattacks pose and the metrics that an NT mechanism usually collects and transmits.

For the reasons explained above, we denote our solution as Traffic-Aware Network Telemetry, or TANT for short. A general overview of the TANT solution is shown in Figure 5.1. As can be seen, the main components of the system at the network elements are a traffic flows classifier and the NT controller, which operates according to the telemetry standard in use. The NT controller determines the granularity of the telemetry data to be transmitted depending on the outcomes of the local traffic classifier. In summary, the main contributions of this work are:

- A flexible framework to achieve efficient NT that can be adapted to a variety of NT schemes regardless their way of operation (in-band or out-of-band).
- The design of a lightweight traffic classifier that does not consider the classical 5-tuple (protocol type, source IP address and port, and destination address and port) to identify different types of traffic.
- A methodology to evaluate and implement inference acceleration of ML algorithms making predictions in real-time scenarios, such as the NT use case presented in this work.

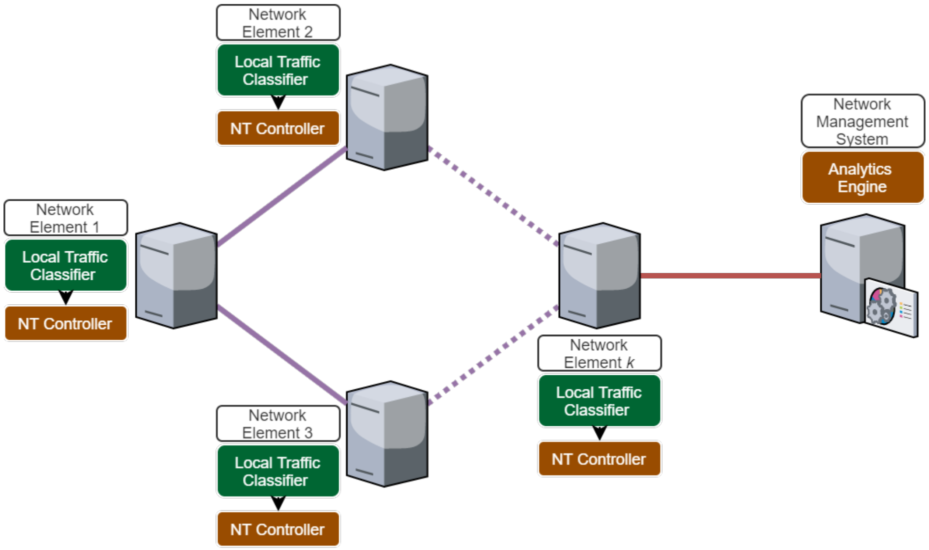


Figure 5.1. TANT system overview. Each network element comprises a traffic classifier and an NT controller, which transmits the NT data to the NT engine.

It is important to point out that, although the TANT framework could be applied in networking setups, such as Wide Area Networks (WAN) and Internet Service Providers (ISPs) networks, its application would be more representative in networks delimited by the local management of a single organization, like enterprises or campuses networks. Also, the implementation of the TANT framework and the utilization of its NT data to tackle inter-domain scenarios' problems, like the one presented in [12], needs further research that is out of the scope of this work.

5.2. Related Work

The challenges that NT poses have been addressed by the research community, in both academic and industry settings, with diverse approaches that generally fell in one of these two main categories: in-band telemetry and out-of-band telemetry. In-band telemetry refers to the case when the NT data transmission usually shares the same link, path, or packet with the users' data whereas the transportation of out-of-band telemetry data does not [120]. The in-band telemetry solutions reviewed in this subsection are related to the In-band Network Telemetry (INT) Dataplane Specification [124]. This specification defines the monitoring system as a system that collects telemetry data sent from different network elements. The components of the monitoring system may be physically distributed but logically centralized. Additionally, with INT the original data packets are monitored and may be modified to carry INT instructions and INT metadata (telemetry data). It is important to highlight that there are other in-band telemetry specifications different from the INT standard. For this reason, we make the distinction between these two terms.

Existing NT systems usually trade off expressiveness (accuracy of the measurements) for scalability (amounts of the telemetry data collected), or vice versa. That is why most of the INT-based schemes aim to reduce the telemetry data transportation overhead and, at the same time, try to avoid losing too much measurement accuracy. Accordingly, authors in [125] present a sampling-based INT mechanism, in which the source node inserts INT headers into the packets at a configurable rate to reduce the overhead. To compensate for the accuracy, their solution also supports a sampling based on events, in which metadata is inserted only when the latency difference between the last hop and the current hop exceeds a predefined threshold. Similarly, Chowdhury *et al.* propose a lightweight INT-based scheme to reduce the overhead by trying to estimate the amount of error that can be introduced at the INT collector if the requested telemetry data are not piggybacked on the current packet [126]. For estimating this error, a predictor function based on Exponentially Weighted Moving Average is used for each telemetry data item of interest.

By encoding the requested data on multiple packets, authors in [127] introduce a probabilistic INT method that bounds the per-packet overhead as low as one bit. The solution supports several aggregation operations that allow efficient encoding of the aggregated data onto packets: per-packet aggregation, static per-flow aggregation, and dynamic per-flow aggregation. Conversely, Wang *et al.* introduce a bandwidth-efficient INT system by tracking the rules matched by the packets of a flow in a previous period [128]. Their proposed solution assigns globally unique IDs to every rules and stores rule-changed INT reports in a database server so that the rate of generated INT reports is reduced. In contrast, [129] considers the overhead not only at the data plane, but also at the control and management planes while employing INT. The authors model the INT Orchestration as an optimization problem and propose two heuristic algorithms to produce feasible solutions in polynomial computational time with respect to the network size and number of flows. From [129], we find interesting the idea of taking into account the three SDN planes to reduce the INT overhead in an orchestrated manner. Finally, Kim *et al.* present a selective INT scheme where an algorithm adjusts the insertion ratio of packets to be monitored according to the frequency of significant changes in network data [130].

What all the solutions reviewed above have in common is the goal to make NT efficient in terms of the usage of the network resources, such as bandwidth and network elements' computational limitations. However, those schemes delimit their applicability to the INT specification, as the per-packet NT data overhead is assumed as the main issue to solve. Although INT is becoming the mainstream telemetry standard, we advocate for a more generalized framework that can also be applied to other in-bound telemetry mechanisms or even out-of-band ones. On the other hand, [125] and [130] are the schemes that relates the most to our proposed framework in terms of the adjustment of the NT data granularity (or rate) to reduce overhead.

5.3. TANT Traffic Classifier

The traffic classification process involves the identification of both normal and different types of abnormal traffic flows. We then design the traffic classifier of our solution using the CICIDS2018 dataset as a benchmark [131]. This and other datasets from the

Canadian Institute for Cybersecurity (CIC) at the University of New Brunswick have been widely used by researchers worldwide to evaluate their network traffic-related methods, such as Internet traffic classification. The CICIDS2018 dataset contains benign and common attacks, which resembles true real-world network data. It also includes the results of the network traffic analysis with labeled flows based on the time stamp, source and destination IP addresses, source and destination ports, and protocols. The dataset was generated with realistic background traffic to profile the abstract behavior of human interactions and includes benign traffic. The final dataset was gathered from different attack scenarios whose attacking infrastructure considers 50 machines and the victim organization has 420 hosts and 30 servers. More than 80 statistical features are extracted from the network traffic in forward and backward directions, as described in [131].

Therefore, the traffic classifier considers multiple classes, including benign traffic and the malicious traffic described by these attacks: DoS-Hulk, DoS-SlowHTTP, DDoS-HOIC, DDoS-LOIC, FTP-BruteForce, and SSH-BruteForce. We chose these attacks because they are the most representative classes in the CICIDS2018 dataset and encompass both TCP and UDP flows. A description of these attacks and the methodology used to obtain their traffic data can be found in [132]. After merging and cleaning the data subsets corresponding to the chosen attacks, the final dataset ended up containing 4,723,155 samples. For the training and test of the traffic classifier, the final dataset is split into 70% and 30%, respectively.

On the other hand, one of our goals is to design a lightweight and protocol-independent scheme to identify network traffic. To achieve so, we first perform an explainable feature engineering process. As we are interested in controlling the granularity of the NT, there is an initial feature selection that considers all time-related features, 27 in total, which are based on traffic flows' metrics (see Table 5.1). It is important to highlight that, in the context of this work, we consider a traffic flow according to the IETF's RFC 7011, Specification of the IP Flow Information Export (IPFIX): "A Flow is defined as a set of packets or frames passing an Observation Point in the network during a certain time interval. All packets belonging to a particular Flow have a set of common properties." [133]. Those common properties include the packet header fields, *i.e.* the 5-tuple of source IP address, destination IP address, source port, destination port, and protocol type. Similarly, we point out that the data used for our analysis and proposed solution correspond to the RFC 7011's definition of Flow Records, which contain measured properties of the flows at the Observation Point. In this way, the features of the input data for the traffic classifier are based on the Flow Records but not on the flows' common properties themselves, such as the 5-tuple.

Table 5.1. Time-related traffic features

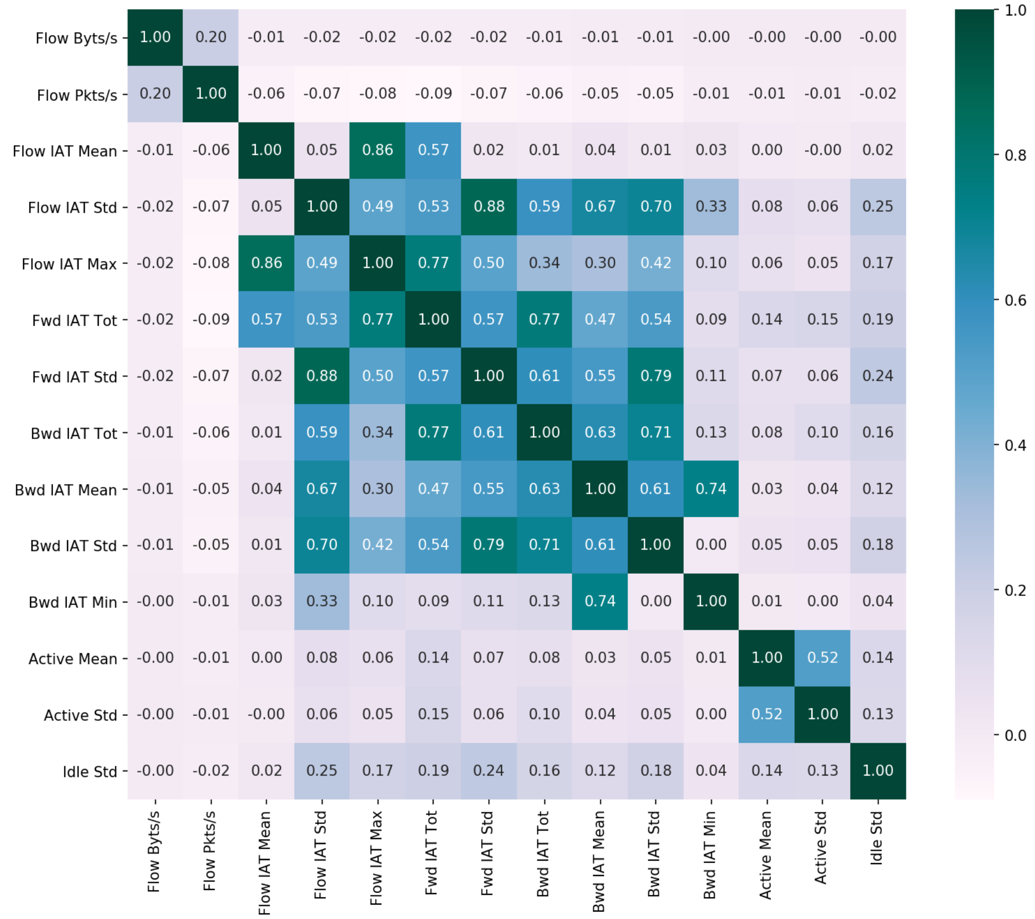
Feature	**Description**
Active Max	Maximum time a flow was active before becoming idle
Active Mean	Mean time a flow was active before becoming idle
Active Min	Minimum time a flow was active before becoming idle
Active Std	Standard deviation time a flow was active before becoming idle
Bwd IAT Max	Maximum time between two packets sent in the backward direction
Bwd IAT Mean	Mean time between two packets sent in the backward direction
Bwd IAT Min	Minimum time between two packets sent in the backward direction
Bwd IAT Std	Standard deviation time between two packets sent in the backward direction
Bwd IAT Total	Total time between two packets sent in the backward direction
Bwd Packets/s	Number of backward packets per second

Flow Byte/s	Number of flow bytes per second
Flow duration	Duration of the flow in microseconds
Flow IAT Max	Maximum time between two packets sent in the flow
Flow IAT Mean	Mean time between two packets sent in the flow
Flow IAT Min	Minimum time between two packets sent in the flow
Flow IAT Std	Standard deviation time between two packets sent in the flow
Flow Packets/s	Number of flow packets per second
Fwd IAT Max	Maximum time between two packets sent in the forward direction
Fwd IAT Mean	Mean time between two packets sent in the forward direction
Fwd IAT Min	Minimum time between two packets sent in the forward direction
Fwd IAT Std	Standard deviation time between two packets sent in the forward direction
Fwd IAT Total	Total time between two packets sent in the forward direction
Fwd Packets/s	Number of forward packets per second
Idle Max	Maximum time a flow was idle before becoming active
Idle Mean	Mean time a flow was idle before becoming active
Idle Min	Minimum time a flow was idle before becoming active
Idle Std	Standard deviation time a flow was idle before becoming active

As a next step in the feature engineering process, we normalized the values of the preselected features and perform a correlation analysis of them. Intuitively, one can suppose that several time-related features described in Table 5.1 are strongly correlated. For example, some of the forward-direction metrics should have a significant correlation with their backward-direction counterparts, the majority of the traffic data correspond to TCP flows. For this reason, we perform another feature selection using the Pearson correlation coefficients. These coefficients are a statistical measure of the linear dependency between two vectors, which are assumed to be normally distributed and to contain n elements each [134]. Thus, the Pearson correlation coefficients are calculated as follows:

$$r(x_1, x_2) = \frac{\sum_{i=1}^{n}\left(x_1^{(i)} - \bar{x}_1\right)\left(x_2^{(i)} - \bar{x}_2\right)}{\sqrt{\sum_{i=1}^{n}\left(x_1^{(i)} - \bar{x}_1\right)^2}\sqrt{\sum_{i=1}^{n}\left(x_2^{(i)} - \bar{x}_2\right)^2}} \qquad (5.1)$$

where x_1 and x_2 are the vectors of the two features being analyzed, $\bar{x}_1$ and $\bar{x}_2$ the mean values of those feature vectors, respectively, and $x_j^{(i)}$ refers to the value of the instance i from feature j. For each coefficient, $r(x_1, x_2) \in [-1, 1]$ and a positive number close to 1 means that an increase or decrease in the values of x_1 is met with the same trend, increase or decrease, in the values of x_2. Accordingly, we discard one of the features whose values have a correlation greater than 0.9 with another feature. The resulting 14 features and their coefficients after carrying out the correlation analysis are shown in Figure 5.2.



	Flow Byts/s	Flow Pkts/s	Flow IAT Mean	Flow IAT Std	Flow IAT Max	Fwd IAT Tot	Fwd IAT Std	Bwd IAT Tot	Bwd IAT Mean	Bwd IAT Std	Bwd IAT Min	Active Mean	Active Std	Idle Std
Flow Byts/s	1.00	0.20	-0.01	-0.02	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	-0.00	-0.00	-0.00	-0.00
Flow Pkts/s	0.20	1.00	-0.06	-0.07	-0.08	-0.09	-0.07	-0.06	-0.05	-0.05	-0.01	-0.01	-0.01	-0.02
Flow IAT Mean	-0.01	-0.06	1.00	0.05	0.86	0.57	0.02	0.01	0.04	0.01	0.03	0.00	-0.00	0.02
Flow IAT Std	-0.02	-0.07	0.05	1.00	0.49	0.53	0.88	0.59	0.67	0.70	0.33	0.08	0.06	0.25
Flow IAT Max	-0.02	-0.08	0.86	0.49	1.00	0.77	0.50	0.34	0.30	0.42	0.10	0.06	0.05	0.17
Fwd IAT Tot	-0.02	-0.09	0.57	0.53	0.77	1.00	0.57	0.77	0.47	0.54	0.09	0.14	0.15	0.19
Fwd IAT Std	-0.02	-0.07	0.02	0.88	0.50	0.57	1.00	0.61	0.55	0.79	0.11	0.07	0.06	0.24
Bwd IAT Tot	-0.01	-0.06	0.01	0.59	0.34	0.77	0.61	1.00	0.63	0.71	0.13	0.08	0.10	0.16
Bwd IAT Mean	-0.01	-0.05	0.04	0.67	0.30	0.47	0.55	0.63	1.00	0.61	0.74	0.03	0.04	0.12
Bwd IAT Std	-0.01	-0.05	0.01	0.70	0.42	0.54	0.79	0.71	0.61	1.00	0.00	0.05	0.05	0.18
Bwd IAT Min	-0.00	-0.01	0.03	0.33	0.10	0.09	0.11	0.13	0.74	0.00	1.00	0.01	0.00	0.04
Active Mean	-0.00	-0.01	0.00	0.08	0.06	0.14	0.07	0.08	0.03	0.05	0.01	1.00	0.52	0.14
Active Std	-0.00	-0.01	-0.00	0.06	0.05	0.15	0.06	0.10	0.04	0.05	0.00	0.52	1.00	0.13
Idle Std	-0.00	-0.02	0.02	0.25	0.17	0.19	0.24	0.16	0.12	0.18	0.04	0.14	0.13	1.00

Figure 5.2. Correlation matrix of the selected features based on the Pearson coefficients.

For the traffic flows classifier, we consider the following classification techniques, which are deemed by ML researchers and practitioners as efficient methods for multi-class problems: Logistic Regression with Stochastic Gradient Descent training (LR-SGD), linear Support Vector Machines with Stochastic Gradient Descent training (SVM-SGD), Random Forest (RF), Extra Trees (ET), Light Gradient Boosting Machine (LightGBM), and Extreme Gradient Boosting (XGBoost). In order to compare the outcomes of these methods, we use the F1-score as the statistical measure of the classification quality, defined by the harmonic mean of the precision and the recall [135], as follows:

$$F_1 = 2\frac{P \cdot R}{P + R} \tag{5.2}$$

where the recall, R, represents the ratio between the number of correct positive results and the number of all relevant samples, and the precision, P, is the relation between the number of correct positive results and the number of positive results. Figure 5.3 shows the comparison of the F1-scores of the abovementioned classifiers before and after performing the feature selection based on the Pearson correlation analysis.

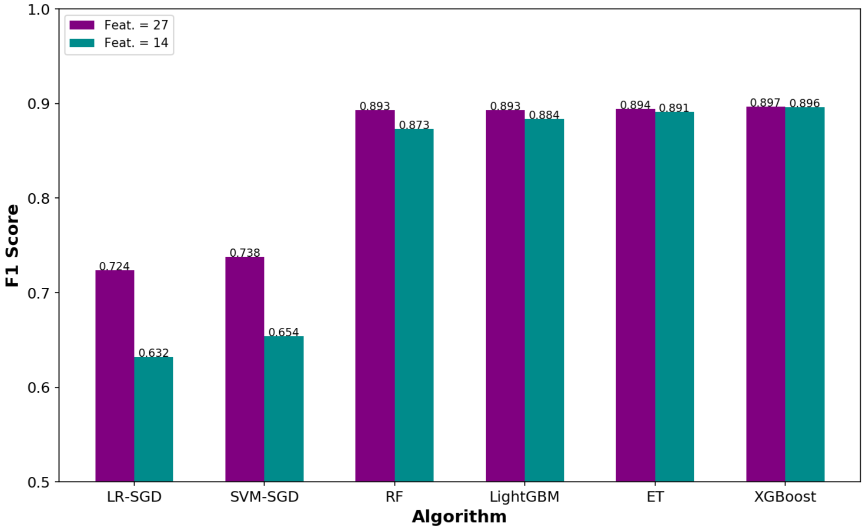


Figure 5.3. Classifiers' scores comparison before and after first feature selection.

As can be seen, the accuracy of the LightGBM, ET, and XGBoost classifiers are slightly lower when almost half of the features (14 out of 27) are used. In contrast, although faster in training, LR-SGD and linear SVM-SGD algorithms are outperformed by the other three in both cases. It is important to highlight that we are more interested in the inference times, rather than the training times, as our goal is to come up with a lightweight traffic classifier to efficiently make predictions in real time. That is why we are not comparing the training times that the algorithms take, however, the inference times will be compared in the performance evaluation of the proposed solution. For the classifier design, we intent to engineer a method that employs a reduced number of features without sacrificing the accuracy too much, so that its complexity is lowered, especially when inferencing traffic anomalies for the NT control process.

As a further step, we complete another feature extraction based on the importance analysis, which helps identify what the most informative features are during the classification process. In this way, we could reduce even more the number of features needed for the inference, if the accuracy is not significantly degraded. We explore this possibility by calculating the Permutation Feature Importance (PFI): a model inspection technique and especially useful for non-linear classifiers. This technique is model agnostic and breaks the relationship between the feature and the target. The PFI for a feature x_j is defined as the average increase in prediction loss, $\mathcal{L}$, when the feature is permuted in training or test dataset [136], as follows:

$$PFI_j = \frac{1}{M}\sum_{m=1}^{M}\frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{L}\left(y^{(i)}, f\left(\hat{x}_j^{m(i)}, x_{\bar{j}}^{(i)}\right)\right) - \mathcal{L}\left(y^{(i)}, f\left(x^{(i)}\right)\right)\right) \tag{5.3}$$

where $f\left(x^{(i)}\right)$ and $y^{(i)}$ refer to the model predictions and the targets, respectively , $\hat{x}_j^{m(i)}$ is a permutation of x_j, M is the number of repeated permutations, and $x_{\bar{j}}$ refers to the complementary feature space. Figure 5.4 shows the PFI coefficients calculated for the classification algorithm with the highest F1-score, *i.e.* XGBoost, over the test subset (meaning $n = 1{,}416{,}947$) and with $M = 15$. As can be seen, the first nine features in importance contribute to over 95% of the classification process.

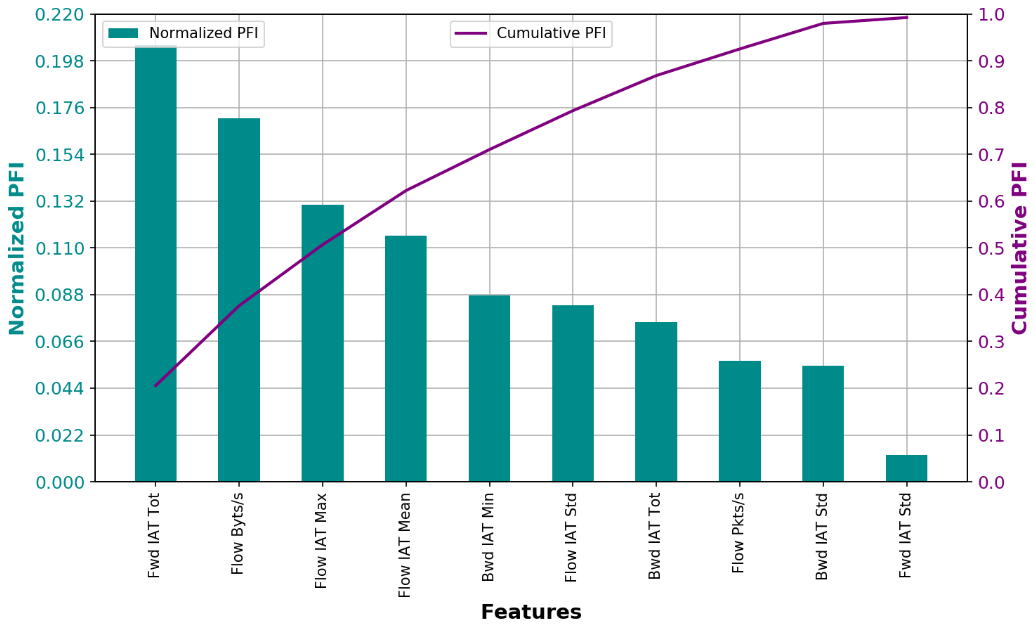


Figure 5.4. Feature ranking based on PFI calculation.

Finally, we test the LightGBM, ET, and XGBoost classifiers only with the top nine features selected from the PFI analysis. The new F1-scores are compared with the previous ones in Table 5.2, which reveals that the changes in the classification accuracy are minuscule, especially for the LightGBM and XGBoost algorithms. This explainable feature space reduction allows the traffic identification process to be less complex and, as a result, to achieve shorter inference times for the NT control mechanism at each network element. The inference performance will be evaluated and discussed in the next subsection. For that evaluation, we compare the best two techniques in terms of accuracy in the reduced feature space, *i.e.* LightGBM and XGBoost, which yield F1-scores of 0.899 and 0.897, respectively, after tuning their hyperparameters [137]. Note that we avoid overturning the hyperparameters that add complexity and make the models more likely to overfit such as the maximum depth and the maximum leaves of the trees. In this way, we keep the structure of both the LightGBM and XGBoost models comparable for the inference performance benchmarking as well as more generalized for making predictions on unseen data.

Table 5.2. F1-Scores comparison after second feature selection

Classifier	**Feat. = 27**	**Feat. = 9**	**Difference**
ET	0.89370	0.89102	0.00268
LightGBM	0.89300	0.89232	0.00068
XGBoost	0.89667	0.89585	0.00082

5.3.1. Inference Acceleration

As explained earlier, the main goal of having a reduced feature space without significantly sacrificing the traffic classification accuracy is to decrease the model's complexity and, therefore, the inference time. In achieving so, the classifier may be implemented in more realistic network scenarios and operate in real time. We go further towards this goal by utilizing an ML inferencing accelerator. For this work, we employ the tools from the Open Neural Network Exchange (ONNX) framework to improve the performance of our model. ONNX is an open ecosystem that provides a standard format for representing the prediction function of trained ML models [138]. It defines an extensible computation graph model and the models trained using several ML frameworks can be exported to ONNX. With ONNX, each computation dataflow graph is structured as a list of nodes that form an acyclic graph, a process known as serialization. As a result, ONNX offers a convenient interoperability of ML models across frameworks and that is why it is widely backed by important companies in the Artificial Intelligence (AI) industry.

We then operationalize the optimized traffic classifier by ONNX with the ONNX Runtime: a high-performance and resource-efficient inference engine for ML models that takes advantage of the specific hardware capabilities where the model is run on [139]. ONNX Runtime can perform inference for any prediction function converted into the ONNX format and its cross-platform nature allows it to be run on different hardware and operating systems. In this manner, ONNX Runtime tries to parallelize the model’s operations and optimizes the model graph by applying graph transformation, that is, elimination and fusion of graph nodes.

Accordingly, we assess the efficiency of the LightGBM and XGBoost classifiers when making predictions for one observation at a time, a common situation in computer networking scenarios such as the use case for this work. To achieve so, we take 15,000 random samples from the resulting dataset after reducing the feature space, as explained in the previous subsection, and measure the processing time that each model takes to predict the type of traffic flow (one sample corresponds to one flow). Similarly, we trace the allocated memory to process each prediction. Figure 5.5 shows the averaged computation times and the averaged RAM usage over the 15,000 samples.

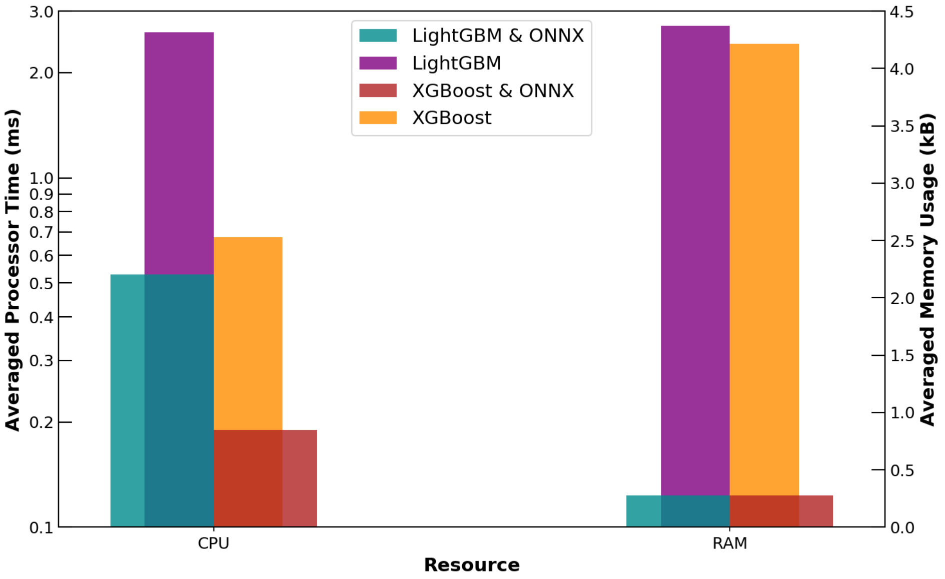


Figure 5.5. Computational resources used by the classifier algorithms for inferring in a single call.

Note the logarithmic scale used for comparing the processing times. As can be seen, XGBoost algorithm achieves faster predictions than LightGBM on batches of one sample in size. More importantly, it is evident that ONNX optimization does accelerate the inference time by a factor of 4.9x and 3.6x for LigthGBM and XGBoost, respectively. Similarly, the memory usage is significantly reduced when ONNX is employed, being nearly the same for both algorithms and improved by a factor of 15.9x, for LigthGBM, and a factor of 15.3x, for XGBoost. All these measurements were obtained by running the ONNX inference calls on a machine with Intel® Xeon® CPU E5-2686, four cores @ 2.30 GHz, and Ubuntu 18.04.4. We also point out that these measured values correspond to the complexity exhibited by a single flow and that complexity grows linearly with the number of flows, N_f, meaning a time complexity of $\mathcal{O}(N_f)$ when multiple flows are considered.

It is important to highlight that several research works have reported lower processing times of LightGBM, although with lower accuracy scores too, when compared to XGBoost. However, LightGBM may be faster when being trained or making batch predictions, but not when inferencing on one observation at a time. This is due to the hyperparameter tuning that the LightGBM needs in order to get similar or higher accuracy than XGBoost. That tuning might result in a more complex model, which significantly affects the inferencing performance. Therefore, we consider the XGBoost algorithm for our proposed efficient telemetry scheme and the performance assessments in the rest of this work. Again, as we discussed in the introduction, we are more interested in attaining a reasonable traffic identification accuracy with an algorithm that provides fast inference in a single call.

5.4. TANT Controller

Network applications require NT to be elastic enough in order to efficiently use the network resources and reduce the performance degradation. Also, routine network monitoring should cover the entire network with low data sampling rate. However, NT data rate may be boosted when issues arise or trends emerge [140]. That is the ultimate goal of the Network Telemetry Controller module in our scheme. As a use case, we evaluate our solution by means of a postcard-like telemetry mechanism, such as the Postcard-Based Telemetry (PBT) or the INT in eXport Data mode (INT-XD). In this mode, INT nodes directly export metadata from their dataplane to the monitoring system based on the INT instructions configured at their Flow Watchlists. A Flow Watchlist is a dataplane table that matches on packet headers and applies INT instructions on each matched flow. The instructions indicate which INT metadata to collect at each INT node and they are either configured at each INT-capable node's Flow Watchlist or written into the INT header. Although INT-XD is a valid mode of operation, it does not represent the classic and the default hop-by-hop INT operation, where the INT devices embed both instructions and metadata, *i.e.* telemetry data, and the packets are modified the most [124].

Similarly, the PBT-M, a packet-marking variation of the PBT, does not require the encapsulation of telemetry instruction headers, avoiding some of the implementation challenges of the instruction-based PBT and the default INT, also known as on-path telemetry in passport mode [141]. PBT-M uses a marking-bit in the existing headers of user packets to trigger the NT data collection and export. If an NT node detects the mark, a postcard (a dedicated packet triggered by a marked user packet) is generated and transmitted to the NT collector. This postcard packet contains the data requested and configured by the management plane. The main advantage of PBT-M is that it avoids growing user packets with new headers and introducing new data plane protocols. However, the data plane devices need to be configured to know what NT data to collect. Another important benefit of PBT-M is that the collected NT data can be transported independently through in-band or out-of-band channels and the types of data collected from each node may be different according to the application requirements and nodes' capabilities. Nevertheless, since each postcard packet has its own header, the overall network bandwidth overhead of PBT may be higher than the passport-based NT, depending on the number of postcards to be transmitted.

For these reasons explained above, our TANT solution is designed as a PBT-M-like scheme that additionally takes into account the granularity of the NT data to be transported, so that the network bandwidth overhead is minimized. To achieve so, we assume that the levels of granularity can be marked through some or all of the 8 bits of the Type of Service (ToS) field of a standard IP packet header. In this way, a network device acting as the NT Source detects the type of traffic that is forwarding and, based on that, marks the level of granularity needed. Then, both NT Source, NT Transit, and NT Sink devices send postcard packets to the NT Monitoring Engine. Finally, the NT Sink unmarks the IP headers. It is important to highlight that, similar to the PBT-M scheme, TANT assumes that the NT devices are instructed on what kind of NT data collect and transmit by the management plane beforehand.

With respect to the granularity levels of the NT data, we analyze the packets' Inter-Arrival Time (IAT) of the types of traffic identified by the classifier. To this end, we explore the values of the attribute describing the average IAT between two packets sent in the forward direction (Fwd IAT Mean) from the whole CICIDS2018 dataset. We point out that, by selecting these IAT values, our analysis is more realistic so that the granularity levels are applicable to real-time scenarios. Also, the selection of the IAT values in the forward direction is consistent with the NT specifications described above, in which a network element triggers the telemetry tasks and forwards the NT instructions to the next elements ahead. On the other hand, the packet IAT values have a significant relationship with the type of traffic that a network element is forwarding and cannot be easily obfuscated or manipulated [142]. For this reason, the IAT characteristics of packets have been used to detect malicious traffic patterns, such as the one described by DDoS attacks [143]. Consequently, we define five levels of NT granularity according to values of the Fwd IAT Mean feature for each traffic class, as shown in Table 5.3. Note that this attribute is not part of the group of nine features finally selected for the proposed traffic classifier (see Figure 5.4).

Table 5.3. Granularity Levels

Granularity (ms)	**Type(s) of Traffic**
100	Benign
10	DDOS attack-HOIC, DoS attacks-Hulk
1	SSH-BruteForce
0.5	DDOS attack-LOIC-UDP
0.1	FTP-BruteForce, DoS attacks-SlowHTTPTest

5.5. Experimentation and Evaluation Results

In this section, we provide the details about the experimentation setup for the evaluation of our proposed solution. The network topology for our experiments is similar to the one presented in [130], although we consider one path only for the NT Transit devices, instead of two, Figure 5.6. The reason why is because their experiments focus on the path changes whereas ours are focused on the variations of the types of traffic. Nevertheless, the ultimate goal is the same: to compare the performance of the proposed NT mechanism against INT when the traffic flows are affected, either by the paths they are being transported on or the specific traffic type they are carrying. Likewise, we use the Mininet emulator as the software tool to implement and evaluate the TANT prototype in the described network topology [119]. Mininet is a suitable tool for our use case, as it allows a flexible SDN environment with high degree of confidence for real-time tests [108].

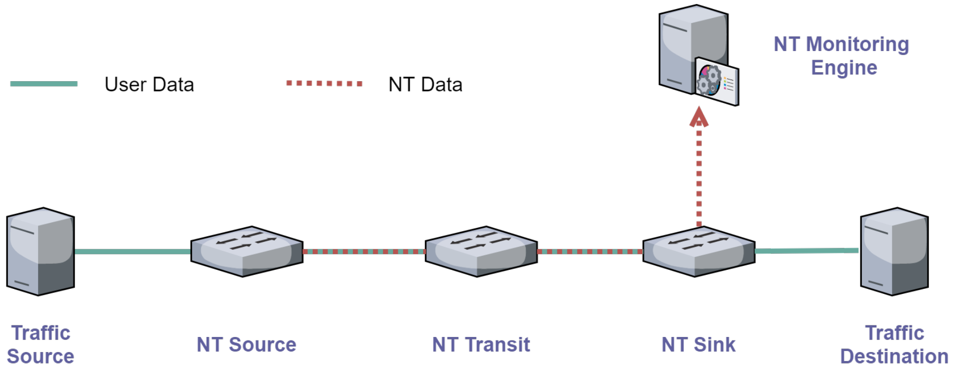


Figure 5.6. Network scenario for the TANT use case evaluation.

On the other hand, we use Scapy [144] as the tool to manipulate the standard IP packets in the TANT implementation. Scapy can be used to construct packets of a variety of existing or new protocols, send and receive them, match requests and replies, and more [145]. Accordingly, the NT Source, NT Transit, and NT Sink devices transmit manipulated PBT-M-like packets to the NT Monitoring Engine, as described in the previous section, by means of Scapy's protocol stacking and fields manipulation functionalities. More specifically, an IP packet is created with the standard IP header and 12 bytes are stacked as the payload of that packet. The rationale behind having a 12-byte payload is that we intend to compare our TANT solution to the conventional INT and the solution presented in [130], which is a scheme based on INT, although with modifications. According to that work, three types of INT metadata are considered as examples for its evaluation: switch ID, hop latency, and queue occupancy. These NT data is inserted into the user data every hop by each network element involved in the INT process, *i.e.* the NT Source, the NT Transit, and the NT Sink. Taking into account the INT specification [124], INT metadata per measured variable occupy 32 bits (4 bytes). Therefore, we consider NT data of 12 bytes in length to make it comparable to the three INT measurements considered in the experiments of [130].

In relation to the evaluation of both TANT modules working together, we run the Mininet emulation as follows: the Traffic Source host picks traffic samples randomly, meaning that any of the types of traffic described in Table 5.3 may be selected. Based on this random selection, the TANT Traffic Classifier determines the type of traffic by considering the selected flow attributes, as explained in Section 5.2: Flow Byts/s, Flow Pkts/s, Flow IAT Mean, Flow IAT Std, Flow IAT Max, Fwd IAT Tot, Bwd IAT Min, Bwd IAT Std, and Bwd IAT Tot. Based on the identified traffic, the NT granularity is established for that flow according to the levels showed in Table 5.3. Afterwards, the NT Source creates an IP packet and uses three of the eight ToS bits to indicate the level of granularity needed, as explained in Section 5.3. Note that three bits are enough to mark all the five granularity levels that the TANT scheme considers for the use case presented in this chapter. However, more bits could be used for that purpose. Additionally, the NT Source inserts the 12-byte payload and send the NT packet to the NT Monitoring Engine. As TANT implements a PBT-like NT, all the NT Transits and the NT Sink perform a similar task in order transmit their NT data to the NT Monitoring Engine. Recall that, similar to the PBT-M specification, we are assuming that all the NT nodes know the measurement data that they need to collect and send a priori. This could be accomplished by means of instructions from the management plane. Finally, each NT node starts transmitting the NT data of the pre-determined measurements in the granularity intervals specified in the ToS bits of the IP packet.

In order to determine the network overhead, we measure the throughput every five seconds using the iPerf tool [146]. More specifically, we set up a pair of monitor hosts in the emulation environment, one of them actively logging the measured throughput by means of sending probe packets to the other one in 5-second intervals. In this way, we measure the throughput without transmitting any NT data and, right after that, the network throughput while the NT data is being transmitted for another 5 s. The network overhead is then calculated by subtracting the measured throughput with NT data from the measured throughput without it. Again, method is similar to the one utilized in [130].

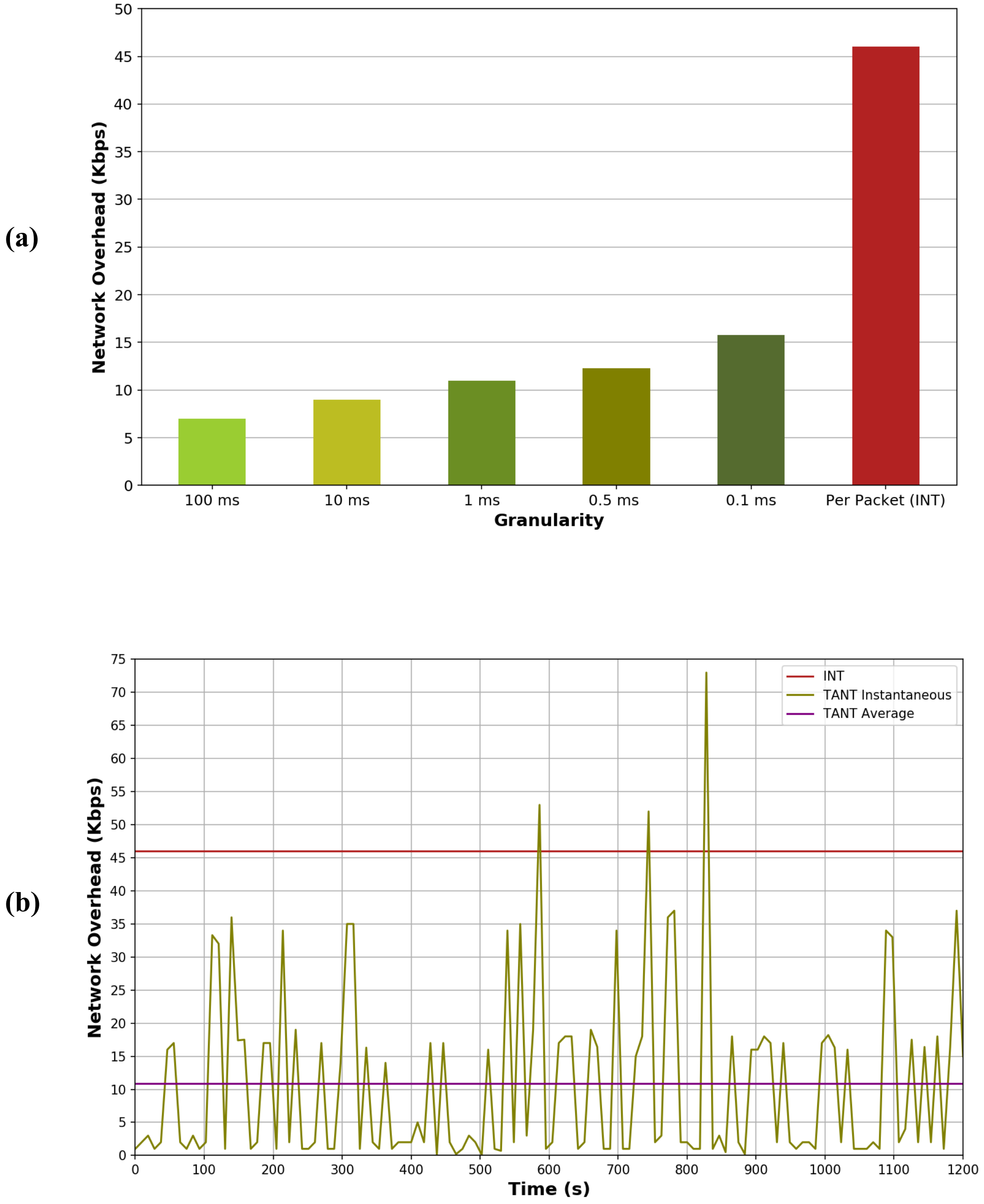


Figure 5.7. Evaluation resutls of TANT and its comparison against the classic INT. a) Network bandwidth overhead reduction per granularity level. b) Instantaneous and average network overhead measured during 1,200 seconds of network emulation.

Figure 5.7 depicts the results of our solution evaluation. As can be seen, the proposed TANT scheme achieves a substantial lower overhead when compared to the regular INT mechanism: the worst-case granularity, *i.e.* 0.1 ms, represents less than 50% of the INT's overhead (Figure 5.7a). With respect to the emulation in real time, TANT attains a reduction of 76.4% in network overhead, on average (Figure 5.7b).

Furthermore, this overhead decrease outperforms the reduction reported in [130], which is 37% less than the conventional INT. It is important to point out that there are some transients of the instantaneous measures of TANT that overpass the INT overhead. These transients are due to the abnormal traffic detected by the classifier, which, at the same time, lowers the granularity. However, our TANT mechanism adaptively changes the granularity of the NT data when normal flows or other types of traffic flows are detected. As a result, the overall network overhead is considerably lesser than that produced by the per-packet INT's granularity.

5.6. Summary

Network Telemetry (NT) is a crucial component in today's networks, as it provides the network managers with important data about the status and behaviour of the network elements. NT data are then utilized to get insights and rapidly take actions to improve the network performance or avoid its degradation. Intuitively, the more data are collected, the better for the network managers. However, the gathering and transportation of excessive NT data might produce an adverse effect, leading to a paradox: the data that are supposed to help actually damage the network performance. This is the motivation to introduce a novel NT framework that dynamically adjusts the rate in which the NT data should be transmitted.

In this chapter, we presented a NT scheme that is traffic-aware, meaning that the network elements collect and send NT data based on the type of traffic that they forward. The evaluation results of our Machine Learning-based mechanism show that it is possible to reduce by over 75% the network bandwidth overhead that a conventional NT scheme produces.

Chapter 6

Conclusion

6.1. Dissertation Conclusions

The main conclusions of this dissertation are summarized as follows:

- The research work of this thesis presented a set of novel methods based on Machine Learning techniques to achieve the realization of the Intelligent Networking Automation paradigm in several scenarios.
- The proposed solutions are data-driven and consist of predictors and agents, which operate at network elements such as routers, switches, or network servers.
- We have effectively applied ML techniques of several types (including supervised, unsupervised, and reinforcement learning) to solve real-world networking challenges.
- We showed that the developed schemes can cope with complex networking situations that involve hard decision-making actions.
- The evaluation of the solutions has been carried out through realistic networking scenarios and with data gathered from either actual network deployments or real-time environments.
- For all the presented frameworks, we have considered standardized network protocols and technologies as use cases. However, they are agnostic enough to be utilized with other mechanisms not considered in this research work.

6.2. Discussion on the Findings and Limitations of this Thesis

In Chapter 2, we proposed a scheme for load balancing in HetNets that can be applied to dense IoT networks such as Smart Cities scenarios. Our approach is based on several ML techniques to discover hidden patterns using PCA, learn from the labeled data by means of supervised probabilistic classifiers, and make decisions through an MDP. As a use case, we validated our method with data from an actual LoRaWAN IoT network. Once we preprocessed the data, we confirmed that such a network deployed in urban areas can be deemed as a HetNet.

We demonstrated that with our scheme the goal of device-BS association biasing can be achieved based on predictions that are made by obviating signal-based features. Unlike other related works, we used labeled data for biasing the device-BS association through a supervised classifier. This approach solves the CRE problem in such a manner that is less complex to implement than other solutions based on reinforcement learning. Therefore, the proposed association biasing method might be more suitable in scenarios where the computational resources of core network elements, such as the Network Server, are more constrained.

We also confirmed that our MDP-based decision-making model for the traffic offloading has better results when the classifier's predictions are considered. The evaluation results describe the improvement of network capabilities in terms of PDR (an increase of 50%) and reduction of ECD (nearly a decrease of 20%). On the other hand, although MDPs are the basis for reinforcement learning algorithms such as Q-learning, our method does not consider the action-value function $Q(s, a)$, i.e., the current state and each possible action that can be taken individually. In other words, in our method the policy and expected reward are based on the current state and the average across all of the actions that can be taken. Therefore, our method needs less data as the function Q is not considered. However, a further study is needed to determine the trade-off of getting better results by including Q and the likely longer time to learn and make decisions. This is also a relevant consideration for wireless networks with very restricted resources.

In this work, we validated our methodology through a specific standard, but the method may be implemented in IoT networks operating under other standards, particularly in dense environments. It is also important to highlight that the only process of our scheme that runs in real time is the MDP and, consequently, the data preprocessing and the classification training can be carried out offline.

Although the time delay caused by the decision process of our model may be unacceptable for several WAN RAT applications, we point out that a particular characteristic of technologies like LoRaWAN is that they are focused on the connectivity of devices that transmit messages in relatively long periods at low data rates. Nevertheless, more research is needed about the optimization of methods like the one we have proposed, as well as the time complexity analysis for their implementation in specific solutions, especially where there is a large number of end devices.

Thinking of a LoRaWAN network, specifically, we point out the importance of considering more adjusted parameters such as data rate, number of retransmissions, and packet arrival rate. Since we used values corresponding to worst-case scenarios in our analytical models, better results can be achieved with our scheme by adjusting those variables to specific situations.

In Chapter 3, we showed how the appropriate tuning of AQM parameters can improve the RTT and throughput of TCP connections in a dynamic IP network. Additionally, we showed that it is possible to take advantage of the ECN mechanism to predict congestion on the rest-of-path. We modeled a congestion predictor based on an LSTM, which we pre-trained with data of an unknown network topology. We exposed how to transfer the predictor model to a new network and get good estimates with a rapid re-training. Also, we described a solution for the decision-making problem on the parameters that an AQM scheme should have according to the networks’ conditions.

We also demonstrated that it can be achieved by modeling the problem as an MDP and finding pair values of state-action through the Q-learning algorithm. Although we employed the power function of the connection as the reward function, our method can work with other rewards depending on the applications or the TCP connection variable to be optimized. However, we point out that the proposed Intelligent AQM has the limitation of working with TCP traffic whose headers can be extracted by the router utilizing the scheme. In other words, if the TCP traffic is being carried through encrypted tunnels, such as those used in Virtual Private Networks, the ECE-marked packets cannot be distinguished by the router.

In Chapter 4, we showed how the appropriate tuning of AQM parameters can improve the RTT and throughput of inter-domain connections. We presented our FIAQM solution, which leverages the characteristics of existing AQM schemes in such scenarios where several parties are involved in a communication process and privacy is a major consideration. The main components of the FIAQM architecture effectively applies the fundamentals of the FL approach to attain congestion control between ASes managed by different organizations and whose network data cannot be shared. We described in detail the main components of FIAQM: an LSTM trained in a federated fashion to predict the beyond-own-domain congestion and an AQM parameter tuner based on the Q-learning algorithm. We also explained how these two components integrate to make possible for a border router to dynamically tune the AQM scheme of its link queue that connects to the border router in another domain.

On the other hand, we evaluated the performance of FIAQM in a realistic environment by means of network emulations. Despite the limitations of the software tool used to this end, our solution can be easily adapted to other real-world environments. It is important to highlight, however, that the way we obtained the drop rate data has some limitations in terms of the routers' operating system functionalities. For example, we used Linux TC tool to collect the queue statistics, whose performance heavily relies on the number of processes that the CPU carries out.

In Chapter 5, we presented a novel framework for efficient collection and transportation of network telemetry data by making the network devices "aware" of the traffic types that they are forwarding. To accomplish so, our TANT scheme comprises two principal modules: an ML-based traffic classifier and an NT controller that adjusts the level of granularity of the telemetry data. We also showed how the inference process of the classifier can be accelerated in order to make per-flow predictions in shorter times and using less memory, important characteristics for any NT mechanism working in real time. In addition, we evaluated the performance of the proposed scheme by means of network emulations and demonstrated that TANT can reduce the network bandwidth overhead to about ¼ of the overhead caused by the classic INT scheme.

Despite its advantages, it is worth mentioning that the main drawback of the TANT framework is that the levels of NT granularity should be known by the nodes beforehand. An automatic and dynamic selection of those levels have not been considered and further research is needed in that regard.

6.3. Future Work

As a future work for our first research problem, an even more realistic scenario may be set such as a prototype network with a server running our scheme and a significantly large number of tiny devices using its services. Additionally, the possibility of combining some of other techniques described in the literature with ours might be explored, to obtain better results in terms of energy efficiency. However, in future implementations we recommend performing these tasks periodically in order to get more accurate results thanks to the updated data.

In terms of the Intelligent AQM scheme, we propose to test our method with different TCP congestion control mechanisms, as well as more AQM algorithms. We also point out that, although our experiments included only two AQM schemes with queue delay as the target parameter, the proposed intelligent method could be easily adapted to other schemes with different target parameters such as the queue size. Finally, although our experiments included only FQ-CoDel as the AQM scheme, the proposed FIAQM method could be straightforwardly implemented with other schemes. In those cases, the only necessary changes would be the redefinition of the set of actions for the AQM tuner module and the inclusion of the specific instructions for the desired AQM scheme configuration in Linux.

With regard to the FIAQM framework, the performance of future implementations may be further improved by considering other design aspects for the neural network of the FCP. For instance, different activation functions could yield more accurate and faster predictions of congestion in situations where shorter time intervals for the measurements are required. Moreover, although in this work we proposed the use of metrics directly taken from the queues as the income data for FIAQM, other kinds of data may easily feed our proposed method. For example, as we mentioned in Section 4.2, the metadata reported by routers employing the INT standard can be adapted to be used in FIAQM. However, how to incorporate INT metrics in multi-domain settings and Machine Learning-based solutions such as FIQAM requires further research.

As a future work for the TANT scheme, it would be interesting to include subcategories of benign traffic for the flow classification process. In this way, other types of traffic can be detected event if they do not correspond to cyber attacks. These subclasses of benevolent, but abnormal, traffic might be very useful to detect and take actions on events that can degrade the network performance. However, quality datasets of real network traces that include those situations need to be generated or made publicly available without compromising private data. Additionally, it would be worth exploring the application of the Federated Learning approach to the traffic classifier in the TANT scheme. In this way, a more scalable solution could be accomplished by decentralizing the learning process and, as a result, a more seamlessly deployment across several local networks or even WANs would be also possible.

References

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.

[2] R. Boutaba *et al.*, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *J. Internet Serv. Appl.*, vol. 9, no. 1, p. 16, Jun. 2018, doi: 10.1186/s13174-018-0087-2.

[3] M. Wang, Y. Cui, X. Wang, S. Xiao, and J. Jiang, “Machine Learning for Networking: Workflow, Advances and Opportunities,” *IEEE Netw.*, vol. 32, no. 2, pp. 92–99, Mar. 2018, doi: 10.1109/MNET.2017.1700200.

[4] J. Lunze and L. Grüne, “Introduction to Networked Control Systems,” in *Control Theory of Digitally Networked Dynamic Systems*, J. Lunze, Ed. Heidelberg: Springer International Publishing, 2014, pp. 1–30. doi: 10.1007/978-3-319-01131-8_1.

[5] Ciena Corporation, “What is Closed-loop Automation?” https://www.blueplanet.com/resources/what-is-closed-loop-automation.html (accessed Jan. 21, 2020).

[6] M. H. Behringer *et al.*, “Autonomic Networking: Definitions and Design Goals,” RFC Editor, RFC 7575, Jun. 2015. doi: 10.17487/RFC7575.

[7] M. H. Behringer, B. E. Carpenter, T. Eckert, L. Ciavaglia, and J. C. Nobre, “A Reference Model for Autonomic Networking,” Internet Engineering Task Force, Internet-Draft draft-ietf-anima-reference-model-10, Nov. 2018. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-anima-reference-model-10

[8] X. Long *et al.*, “Autonomic Networking: Architecture Design and Standardization,” *IEEE Internet Comput.*, vol. 21, no. 5, pp. 48–53, 2017, doi: 10.1109/MIC.2017.3481338.

[9] ETSI, “Autonomic network engineering for the self-managing Future Internet (AFI); Generic Autonomic Network Architecture; Part 2: An Architectural Reference Model for Autonomic Networking, Cognitive Networking and Self-Management,” Technical Specification ETSI TS 103 195-2 V1.1.1, May 2018.

[10] ETSI, “Zero-touch network and Service Management (ZSM); Reference Architecture,” Group Specification ETSI GS ZSM 002 V1.1.1, Aug. 2019.

[11] H. Hawilo, M. Jammal, and A. Shami, “Network Function Virtualization-Aware Orchestrator for Service Function Chaining Placement in the Cloud,” *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 643–655, Mar. 2019, doi: 10.1109/JSAC.2019.2895226.

[12] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, “Software defined networking: State of the art and research challenges,” *Comput. Netw.*, vol. 72, pp. 74–98, Oct. 2014, doi: 10.1016/j.comnet.2014.07.004.

[13] M. Abu Sharkh, A. Ouda, and A. Shami, “A resource scheduling model for cloud computing data centers,” in *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Jul. 2013, pp. 213–218. doi: 10.1109/IWCMC.2013.6583561.

[14] M. Injadat, F. Salo, A. B. Nassif, A. Essex, and A. Shami, “Bayesian Optimization with Machine Learning Algorithms Towards Anomaly Detection,” in *2018 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2018, pp. 1–6. doi: 10.1109/GLOCOM.2018.8647714.

[15] M. Injadat, A. Moubayed, A. B. Nassif, and A. Shami, “Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection,” *IEEE Trans.*

Netw. Serv. Manag., vol. 18, no. 2, pp. 1803–1816, Jun. 2021, doi: 10.1109/TNSM.2020.3014929.

[16] ITU-T, “Unified Architecture for Machine Learning in 5G and Future Networks,” Technical Specification FG-ML5G-ARC5G, Jan. 2019.

[17] T. Yu, X. Wang, and A. Shami, “A Novel Fog Computing Enabled Temporal Data Reduction Scheme in IoT Systems,” in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec. 2017, pp. 1–5. doi: 10.1109/GLOCOM.2017.8253941.

[18] T. Yu, X. Wang, and A. Shami, “Recursive Principal Component Analysis-Based Data Outlier Detection and Sensor Data Aggregation in IoT Systems,” *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2207–2216, Dec. 2017, doi: 10.1109/JIOT.2017.2756025.

[19] A. Moubayed, A. Shami, and H. Lutfiyya, “Wireless Resource Virtualization With Device-to-Device Communication Underlaying LTE Network,” *IEEE Trans. Broadcast.*, vol. 61, no. 4, pp. 734–740, Dec. 2015, doi: 10.1109/TBC.2015.2492458.

[20] A. Ghosh *et al.*, “Heterogeneous cellular networks: From theory to practice,” *IEEE Commun. Mag.*, vol. 50, no. 6, pp. 54–64, Jun. 2012, doi: 10.1109/MCOM.2012.6211486.

[21] J. G. Andrews, “Seven ways that HetNets are a cellular paradigm shift,” *IEEE Commun. Mag.*, vol. 51, no. 3, pp. 136–144, Mar. 2013, doi: 10.1109/MCOM.2013.6476878.

[22] J. G. Andrews, S. Singh, Q. Ye, X. Lin, and H. S. Dhillon, “An overview of load balancing in hetnets: old myths and open problems,” *IEEE Wirel. Commun.*, vol. 21, no. 2, pp. 18–25, Apr. 2014, doi: 10.1109/MWC.2014.6812287.

[23] D. Liu *et al.*, “User Association in 5G Networks: A Survey and an Outlook,” *IEEE Commun. Surv. Tutor.*, vol. 18, no. 2, pp. 1018–1044, Second quarter 2016, doi: 10.1109/COMST.2016.2516538.

[24] A. H. Arani, M. J. Omidi, A. Mehbodniya, and F. Adachi, “A distributed learning–based user association for heterogeneous networks,” *Trans. Emerg. Telecommun. Technol.*, vol. 28, no. 11, p. e3192, Nov. 2017, doi: 10.1002/ett.3192.

[25] M. Kamel, W. Hamouda, and A. Youssef, “Ultra-Dense Networks: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 18, no. 4, pp. 2522–2545, Fourth quarter 2016, doi: 10.1109/COMST.2016.2571730.

[26] J. de C. Silva, J. J. P. C. Rodrigues, A. M. Alberti, P. Solic, and A. L. L. Aquino, “LoRaWAN - A low power WAN protocol for Internet of Things: A review and opportunities,” in *2017 2nd International Multidisciplinary Conference on Computer and Energy Science (SpliTech)*, Jul. 2017, pp. 1–6.

[27] U. Raza, P. Kulkarni, and M. Sooriyabandara, “Low Power Wide Area Networks: An Overview,” *IEEE Commun. Surv. Tutor.*, vol. 19, no. 2, pp. 855–873, Secondquarter 2017, doi: 10.1109/COMST.2017.2652320.

[28] M. Centenaro, L. Vangelista, A. Zanella, and M. Zorzi, “Long-range communications in unlicensed bands: the rising stars in the IoT and smart city scenarios,” *IEEE Wirel. Commun.*, vol. 23, no. 5, pp. 60–67, Oct. 2016, doi: 10.1109/MWC.2016.7721743.

[29] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, “Understanding the Limits of LoRaWAN,” *IEEE Commun. Mag.*, vol. 55, no. 9, pp. 34–40, Sep. 2017, doi: 10.1109/MCOM.2017.1600613.

[30] T. Kudo and T. Ohtsuki, “Cell range expansion using distributed Q-learning in heterogeneous networks,” *EURASIP J. Wirel. Commun. Netw.*, vol. 2013, no. 1, p. 61, Dec. 2013, doi: 10.1186/1687-1499-2013-61.

[31] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews, "User Association for Load Balancing in Heterogeneous Cellular Networks," *IEEE Trans. Wirel. Commun.*, vol. 12, no. 6, pp. 2706–2716, Jun. 2013, doi: 10.1109/TWC.2013.040413.120676.

[32] H. Jiang, Z. Pan, N. Liu, X. You, and T. Deng, "Gibbs-Sampling-Based CRE Bias Optimization Algorithm for Ultradense Networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 1334–1350, Feb. 2017, doi: 10.1109/TVT.2016.2560900.

[33] M. Afshang and H. S. Dhillon, "Poisson Cluster Process Based Analysis of HetNets With Correlated User and Base Station Locations," *IEEE Trans. Wirel. Commun.*, vol. 17, no. 4, pp. 2417–2431, Apr. 2018, doi: 10.1109/TWC.2018.2794983.

[34] J. B. Park and K. S. Kim, "Load-Balancing Scheme With Small-Cell Cooperation for Clustered Heterogeneous Cellular Networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 1, pp. 633–649, Jan. 2018, doi: 10.1109/TVT.2017.2748570.

[35] S. Fan, H. Tian, and W. Wang, "Joint Effect of User Activity Sensing and Biased Cell Association in Energy Efficient HetNets," *IEEE Commun. Lett.*, vol. 20, no. 10, pp. 1999–2002, Oct. 2016, doi: 10.1109/LCOMM.2016.2571287.

[36] K. Yang, L. Wang, S. Wang, and X. Zhang, "Optimization of Resource Allocation and User Association for Energy Efficiency in Future Wireless Networks," *IEEE Access*, vol. 5, pp. 16469–16477, 2017, doi: 10.1109/ACCESS.2017.2722007.

[37] F. Muhammad, Z. H. Abbas, and F. Y. Li, "Cell Association With Load Balancing in Nonuniform Heterogeneous Cellular Networks: Coverage Probability and Rate Analysis," *IEEE Trans. Veh. Technol.*, vol. 66, no. 6, pp. 5241–5255, Jun. 2017, doi: 10.1109/TVT.2016.2614696.

[38] Y. Sun, W. Xia, S. Zhang, Y. Wu, T. Wang, and Y. Fang, "Energy Efficient Pico Cell Range Expansion and Density Joint Optimization for Heterogeneous

Networks with eICIC," *Sensors*, vol. 18, no. 3, p. 762, Mar. 2018, doi: 10.3390/s18030762.

[39] LoRa Alliance Technical Committee, "LoRaWAN^{TM} 1.1 Specification." Oct. 11, 2017.

[40] LoRa Alliance Technical Marketing Group, "LoRaWAN, What is it? A technical review of LoRa and LoRaWAN." Nov. 2015.

[41] "Building a global internet of things network together," *The Things Network*. https://www.thethingsnetwork.org/ (accessed Jul. 16, 2018).

[42] "LoRaWAN Gateways," *Tektelic*. https://tektelic.com/iot/lorawan-gateways/ (accessed Aug. 07, 2018).

[43] "TTN Mapper." https://ttnmapper.org/ (accessed Jul. 17, 2018).

[44] C. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006. Accessed: Mar. 28, 2018. [Online]. Available: //www.springer.com/gp/book/9780387310732

[45] S. Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*. CRC Press, 2015.

[46] "Network Architecture," *The Things Network*. https://www.thethingsnetwork.org/docs/network/architecture.html (accessed Jul. 25, 2018).

[47] A. Pop, U. Raza, P. Kulkarni, and M. Sooriyabandara, "Does Bidirectional Traffic Do More Harm Than Good in LoRaWAN Based LPWA Networks?," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec. 2017, pp. 1–6. doi: 10.1109/GLOCOM.2017.8254509.

[48] S. Rogers and M. Girolami, *A first course in machine learning*. Boca Raton, FL: CRC Press, 2012.

[49] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[50] M. Sokolova and G. Lapalme, “A systematic analysis of performance measures for classification tasks,” *Inf. Process. Manag.*, vol. 45, no. 4, pp. 427–437, Jul. 2009, doi: 10.1016/j.ipm.2009.03.002.

[51] M. van Otterlo and M. Wiering, “Reinforcement Learning and Markov Decision Processes,” in *Reinforcement Learning*, Springer, Berlin, Heidelberg, 2012, pp. 3–42. doi: 10.1007/978-3-642-27645-3_1.

[52] T. J. Sheskin, *Markov chains and decision processes for engineers and managers*. Boca Raton, FL: CRC Press, 2011.

[53] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Pearson Higher Ed, 2012.

[54] B. Reynders, Q. Wang, P. Tuset-Peiro, X. Vilajosana, and S. Pollin, “Improving Reliability and Scalability of LoRaWANs Through Lightweight Scheduling,” *IEEE Internet Things J.*, vol. 5, no. 3, pp. 1830–1842, Jun. 2018, doi: 10.1109/JIOT.2018.2815150.

[55] M. C. Bor, U. Roedig, T. Voigt, and J. M. Alonso, “Do LoRa Low-Power Wide-Area Networks Scale?,” in *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, New York, NY, USA, 2016, pp. 59–67. doi: 10.1145/2988287.2989163.

[56] L. Casals, B. Mir, R. Vidal, and C. Gomez, “Modeling the Energy Performance of LoRaWAN,” *Sensors*, vol. 17, no. 10, p. 2364, Oct. 2017, doi: 10.3390/s17102364.

[57] “Python Data Analysis Library — pandas.” https://pandas.pydata.org/index.html (accessed Jul. 27, 2018).

[58] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *J. Mach. Learn. Res.*, vol. 12, p. 2825−2830, Oct. 2011.

[59] I. Chadès, G. Chapron, M.-J. Cros, F. Garcia, and R. Sabbadin, “MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems,” *Ecography*, vol. 37, no. 9, pp. 916–920, Sep. 2014, doi: 10.1111/ecog.00888.

[60] J. Gettys and K. Nichols, “Bufferbloat: Dark Buffers in the Internet,” *Queue*, vol. 9, no. 11, p. 40:40-40:54, Nov. 2011, doi: 10.1145/2063166.2071893.

[61] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEEACM Trans. Netw.*, vol. 1, no. 4, pp. 397–413, Aug. 1993, doi: 10.1109/90.251892.

[62] K. Nichols and V. Jacobson, “Controlling Queue Delay,” *Queue*, vol. 10, no. 5, p. 20:20-20:34, May 2012, doi: 10.1145/2208917.2209336.

[63] R. Pan, P. Natarajan, F. Baker, and G. White, *Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem*. RFC Editor, 2017. doi: 10.17487/RFC8033.

[64] T. Hoeiland-Joergensen, P. McKenney, D. Taht, J. Gettys, and E. Dumazet, *The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm*. RFC Editor, 2018. doi: 10.17487/RFC8290.

[65] G. Fairhurst and M. Welzl, *The Benefits of Using Explicit Congestion Notification (ECN)*. RFC Editor, 2017. doi: 10.17487/RFC8087.

[66] A. M. Mandalari, A. Lutu, B. Briscoe, M. Bagnulo, and O. Alay, “Measuring ECN++: Good News for ++, Bad News for ECN over Mobile,” *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 180–186, Mar. 2018, doi: 10.1109/MCOM.2018.1700739.

[67] S. Floyd, D. K. K. Ramakrishnan, and D. L. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*. RFC Editor, 2001. doi: 10.17487/RFC3168.

[68] M. Kühlewind and R. Scheffenegger, *TCP Modifications for Congestion Exposure (ConEx)*. RFC Editor, 2016. doi: 10.17487/RFC7786.

[69] F. Li *et al.*, “A comparative simulation study of TCP/AQM systems for evaluating the potential of neuron-based AQM schemes,” *J. Netw. Comput. Appl.*, vol. 41, pp. 274–299, May 2014, doi: 10.1016/j.jnca.2014.01.005.

[70] S. K. Bisoy and P. K. Pattnaik, “An AQM Controller Based on Feed-Forward Neural Networks for Stable Internet,” *Arab. J. Sci. Eng.*, vol. 43, no. 8, pp. 3993–4004, Aug. 2018, doi: 10.1007/s13369-017-2767-9.

[71] N. Bouacida and B. Shihada, “Practical and Dynamic Buffer Sizing using LearnQueue,” *IEEE Trans. Mob. Comput.*, pp. 1–1, Sep. 2018, doi: 10.1109/TMC.2018.2868670.

[72] S. Latré, W. V. de Meerssche, D. Deschrijver, D. Papadimitriou, T. Dhaene, and F. D. Turck, “A cognitive accountability mechanism for penalizing misbehaving ECN-based TCP stacks,” *Int. J. Netw. Manag.*, vol. 23, no. 1, pp. 16–40, 2013, doi: 10.1002/nem.1806.

[73] T. D. Wallace, K. A. Meerja, and A. Shami, “On-demand scheduling for concurrent multipath transfer using the stream control transmission protocol,” *J. Netw. Comput. Appl.*, vol. 47, pp. 11–22, Jan. 2015, doi: 10.1016/j.jnca.2014.09.008.

[74] A. Graves, “Long Short-Term Memory,” in *Supervised Sequence Labelling with Recurrent Neural Networks*, A. Graves, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 37–45. doi: 10.1007/978-3-642-24797-2_4.

[75] Q. Zhang, H. Wang, J. Dong, G. Zhong, and X. Sun, “Prediction of Sea Surface Temperature Using Long Short-Term Memory,” *IEEE Geosci. Remote Sens. Lett.*, vol. 14, no. 10, pp. 1745–1749, Oct. 2017, doi: 10.1109/LGRS.2017.2733548.

[76] J. Ke and X. Liu, “Empirical Analysis of Optimal Hidden Neurons in Neural Network Modeling for Stock Prediction,” in *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, Dec. 2008, vol. 2, pp. 828–832. doi: 10.1109/PACIIA.2008.363.

[77] F. Baker and G. Fairhurst, *IETF Recommendations Regarding Active Queue Management*. RFC Editor, 2015. doi: 10.17487/RFC7567.

[78] C. A. Gomez, A. Shami, and X. Wang, “Machine Learning Aided Scheme for Load Balancing in Dense IoT Networks,” *Sensors*, vol. 18, no. 11, p. 3779, Nov. 2018, doi: 10.3390/s18113779.

[79] R. S. Sutton and A. G. Barto, “Temporal-Difference Learning,” in *Reinforcement Learning: An Introduction*, MIT Press, 2018, pp. 131–132.

[80] G. Appenzeller, I. Keslassy, and N. McKeown, “Sizing router buffers,” in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, Portland, Oregon, USA, Aug. 2004, pp. 281–292. doi: 10.1145/1015467.1015499.

[81] D. Wischik and N. McKeown, “Part I: buffer sizes for core routers,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 75–78, Jul. 2005, doi: 10.1145/1070873.1070884.

[82] Center for Applied Internet Data Analysis, “CAIDA Internet Data -- Passive Data Sources.” http://www.caida.org/data/passive/index.xml (accessed Jan. 07, 2019).

[83] S. Ackerman, E. Farchi, O. Raz, M. Zalmanovici, and P. Dube, “Detection of data drift and outliers affecting machine learning model performance over time,” Dec. 2020, Accessed: Aug. 26, 2021. [Online]. Available: https://arxiv.org/abs/2012.09258v2

[84] “Intelligent method to be used with AQM schemes such as CoDel and FQ-CoDel.” https://github.com/cgomezsu/IntelligentAQM (accessed May 01, 2019).

[85] C. A. Gomez, X. Wang, and A. Shami, “Intelligent Active Queue Management Using Explicit Congestion Notification,” in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013475.

[86] M. Chiesa *et al.*, “Inter-domain networking innovation on steroids: empowering ixps with SDN capabilities,” *IEEE Commun. Mag.*, vol. 54, no. 10, pp. 102–108, Oct. 2016, doi: 10.1109/MCOM.2016.7588277.

[87] K. C. Claffy, D. D. Clark, S. Bauer, and A. D. Dhamdhere, “Policy Challenges in Mapping Internet Interdomain Congestion,” Social Science Research Network, Rochester, NY, SSRN Scholarly Paper ID 2756868, Aug. 2016. Accessed: Jan. 22, 2020. [Online]. Available: https://papers.ssrn.com/abstract=2756868

[88] P. Kairouz *et al.*, “Advances and Open Problems in Federated Learning,” *ArXiv191204977 Cs Stat*, Dec. 2019, Accessed: Jan. 06, 2020. [Online]. Available: http://arxiv.org/abs/1912.04977

[89] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, “Communication-Efficient Learning of Deep Networks from Decentralized Data,” in *Artificial Intelligence and Statistics*, Apr. 2017, pp. 1273–1282. Accessed: Jan. 07, 2020. [Online]. Available: http://proceedings.mlr.press/v54/mcmahan17a.html

[90] D. M. Manias and A. Shami, “Making a Case for Federated Learning in the Internet of Vehicles and Intelligent Transportation Systems,” *IEEE Netw.*, vol. 35, no. 3, pp. 88–94, May 2021, doi: 10.1109/MNET.011.2000552.

[91] D. M. Manias and A. Shami, “The Need for Advanced Intelligence in NFV Management and Orchestration,” *IEEE Netw.*, vol. 35, no. 1, pp. 365–371, Jan. 2021, doi: 10.1109/MNET.011.2000373.

[92] Q. Yang, Y. Liu, T. Chen, and Y. Tong, “Federated Machine Learning: Concept and Applications,” *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12:1-12:19, Jan. 2019, doi: 10.1145/3298981.

[93] YangQiang, LiuYang, ChenTianjian, and TongYongxin, “Federated Machine Learning,” *ACM Trans. Intell. Syst. Technol. TIST*, Jan. 2019, Accessed: Jan. 08, 2020. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3298981

[94] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, “Federated Learning: Challenges, Methods, and Future Directions,” *ArXiv190807873 Cs Stat*, Aug. 2019, Accessed: Jan. 06, 2020. [Online]. Available: http://arxiv.org/abs/1908.07873

[95] Y. Yang *et al.*, “Inter-domain routing bottlenecks and their aggravation,” *Comput. Netw.*, vol. 162, p. 106839, Oct. 2019, doi: 10.1016/j.comnet.2019.06.017.

[96] Y. Zhao, A. Saeed, M. Ammar, and E. Zegura, “Unison: Enabling Content Provider/ISP Collaboration using a vSwitch Abstraction,” in *2019 IEEE 27th International Conference on Network Protocols (ICNP)*, Oct. 2019, pp. 1–11. doi: 10.1109/ICNP.2019.8888032.

[97] W. J. A. Silva, “An Architecture to Manage Incoming Traffic of Inter-Domain Routing Using OpenFlow Networks,” *Information*, vol. 9, no. 4, p. 92, Apr. 2018, doi: 10.3390/info9040092.

[98] A. Dethise, M. Chiesa, and M. Canini, “Privacy-Preserving Detection of Inter-Domain SDN Rules Overlaps,” in *Proceedings of the SIGCOMM Posters and Demos*, Los Angeles, CA, USA, Aug. 2017, pp. 6–8. doi: 10.1145/3123878.3131967.

[99] K. D. Joshi and K. Kataoka, “PRIME-Q: Privacy Aware End-to-End QoS Framework in Multi-Domain SDN,” in *2019 IEEE Conference on Network Softwarization (NetSoft)*, Jun. 2019, pp. 169–177. doi: 10.1109/NETSOFT.2019.8806645.

[100] M. S. Kang and V. D. Gligor, “Routing Bottlenecks in the Internet: Causes, Exploits, and Countermeasures,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, Scottsdale, Arizona, USA, Nov. 2014, pp. 321–333. doi: 10.1145/2660267.2660299.

[101] X. Chen *et al.*, “Fine-grained queue measurement in the data plane,” in *Proceedings of the 15th International Conference on Emerging Networking*

Experiments And Technologies, Orlando, Florida, Dec. 2019, pp. 15–29. doi: 10.1145/3359989.3365408.

[102] Y. Li *et al.*, "HPCC: high precision congestion control," in *Proceedings of the ACM Special Interest Group on Data Communication*, Beijing, China, Aug. 2019, pp. 44–58. doi: 10.1145/3341302.3342085.

[103] S. Skansi, "Recurrent Neural Networks," in *Introduction to Deep Learning: From Logical Calculus to Artificial Intelligence*, S. Skansi, Ed. Cham: Springer International Publishing, 2018, pp. 135–152. doi: 10.1007/978-3-319-73004-2_7.

[104] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "Recurrent Neural Network Architectures," in *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*, F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, Eds. Cham: Springer International Publishing, 2017, pp. 23–29. doi: 10.1007/978-3-319-70338-1_3.

[105] F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, "Properties and Training in Recurrent Neural Networks," in *Recurrent Neural Networks for Short-Term Load Forecasting: An Overview and Comparative Analysis*, F. M. Bianchi, E. Maiorino, M. C. Kampffmeyer, A. Rizzi, and R. Jenssen, Eds. Cham: Springer International Publishing, 2017, pp. 9–21. doi: 10.1007/978-3-319-70338-1_2.

[106] "6.1 Issues in Resource Allocation — Computer Networks: A Systems Approach Version 6.2-dev documentation." https://book.systemsapproach.org/congestion/issues.html (accessed Oct. 09, 2020).

[107] S. Floyd, *Metrics for the Evaluation of Congestion Control Mechanisms*. RFC Editor, 2008. doi: 10.17487/RFC5166.

[108] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM*

Workshop on Hot Topics in Networks, New York, NY, USA, Oct. 2010, pp. 1–6. doi: 10.1145/1868447.1868466.

[109] C. A. Gomez, "FIAQM." https://github.com/cgomezsu/FIAQM (accessed Oct. 05, 2020).

[110] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8026–8037. Accessed: Oct. 05, 2020. [Online]. Available: http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[111] "TensorFlow Federated." https://www.tensorflow.org/federated (accessed May 06, 2020).

[112] "Care and Feeding of Netperf 2.7.X." https://hewlettpackard.github.io/netperf/doc/netperf.html#Global-Options (accessed May 13, 2020).

[113] "Realtime Response Under Load (RRUL) Specification - Bufferbloat.net." https://www.bufferbloat.net/projects/bloat/wiki/RRUL_Spec/ (accessed Jul. 13, 2020).

[114] "tc(8) - Linux Traffic Control Manual Page." https://man7.org/linux/man-pages/man8/tc.8.html (accessed May 05, 2020).

[115] J. Galbraith and O. Saarenmaa, "SSH File Transfer Protocol," Internet Engineering Task Force, Internet-Draft draft-ietf-secsh-filexfer-13, Jul. 2006. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-secsh-filexfer-13

[116] K. Bonawitz *et al.*, "Practical Secure Aggregation for Privacy-Preserving Machine Learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, Oct. 2017, pp. 1175–1191. doi: 10.1145/3133956.3133982.

[117] M. Ion *et al.*, "On Deploying Secure Computing: Private Intersection-Sum-with-Cardinality," presented at the 2020 IEEE European Symposium on Security and Privacy (EuroS&P), Aug. 2020.

[118] S. Feng and H. Yu, "Multi-Participant Multi-Class Vertical Federated Learning," *ArXiv200111154 Cs Stat*, Jan. 2020, Accessed: Sep. 29, 2020. [Online]. Available: http://arxiv.org/abs/2001.11154

[119] "Mininet Overview - Mininet." http://mininet.org/overview/ (accessed May 06, 2021).

[120] L. Tan *et al.*, "In-band Network Telemetry: A Survey," *Comput. Netw.*, vol. 186, p. 107763, Feb. 2021, doi: 10.1016/j.comnet.2020.107763.

[121] A. Morton, "Active and Passive Metrics and Methods (with Hybrid Types In-Between)," RFC Editor, 7799, May 2016. doi: 10.17487/RFC7799.

[122] M. Yu, "Network telemetry: towards a top-down approach," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 49, no. 1, pp. 11–17, Feb. 2019, doi: 10.1145/3314212.3314215.

[123] N. Ouroua, W. Bouzegza, and M. Ioualalen, "Formal Modeling and Performance Evaluation of Network's Server Under SYN/TCP Attack," in *Mobile, Secure, and Programmable Networking*, Cham, 2017, pp. 74–87. doi: 10.1007/978-3-319-67807-8_6.

[124] "In-band Network Telemetry (INT) Dataplane Specification," The P4.org Applications Working Group, Version 2.1, Oct. 2020. Accessed: Feb. 11, 2021. [Online]. Available: https://github.com/p4lang/p4-applications/tree/master/docs

[125] D. Suh, S. Jang, S. Han, S. Pack, and X. Wang, "Flexible sampling-based in-band network telemetry in programmable data plane," *ICT Express*, vol. 6, no. 1, pp. 62–65, Mar. 2020, doi: 10.1016/j.icte.2019.08.005.

[126] S. R. Chowdhury, R. Boutaba, and J. François, "LINT: Accuracy-adaptive and Lightweight In-band Network Telemetry".

[127] R. Ben Basat, S. Ramanathan, Y. Li, G. Antichi, M. Yu, and M. Mitzenmacher, "PINT: Probabilistic In-band Network Telemetry," 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*, New York, NY, USA, Jul. 2020, pp. 662–680. doi: 10.1145/3387514.3405894.

[128] S.-Y. Wang, Y.-R. Chen, J.-Y. Li, H.-W. Hu, J.-A. Tsai, and Y.-B. Lin, "A Bandwidth-Efficient INT System for Tracking the Rules Matched by the Packets of a Flow," in *2019 IEEE Global Communications Conference (GLOBECOM)*, Dec. 2019, pp. 1–6. doi: 10.1109/GLOBECOM38437.2019.9013581.

[129] J. A. Marques, M. C. Luizelli, R. I. Tavares da Costa Filho, and L. P. Gaspary, "An optimization-based approach for efficient network monitoring using in-band network telemetry," *J. Internet Serv. Appl.*, vol. 10, no. 1, p. 12, Jun. 2019, doi: 10.1186/s13174-019-0112-0.

[130] Y. Kim, D. Suh, and S. Pack, "Selective In-band Network Telemetry for Overhead Reduction," in *2018 IEEE 7th International Conference on Cloud Networking (CloudNet)*, Oct. 2018, pp. 1–3. doi: 10.1109/CloudNet.2018.8549351.

[131] "IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB." https://www.unb.ca/cic/datasets/ids-2018.html (accessed Mar. 03, 2021).

[132] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," in *ICISSp*, 2018, pp. 108–116.

[133] P. Aitken, B. Claise, and B. Trammell, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information," RFC Editor, RFC 7011, Sep. 2013. doi: 10.17487/RFC7011.

[134] M. Braschler, T. Stadelmann, and K. Stockinger, *Applied Data Science: Lessons Learned for the Data-Driven Business*. Springer International Publishing, 2019.

[135] O. Campesato, *Artificial Intelligence, Machine Learning, and Deep Learning*. Mercury Learning & Information, 2020.

[136] C. Molnar, G. König, B. Bischl, and G. Casalicchio, "Model-agnostic Feature Importance and Effects with Dependent Features -- A Conditional Subgroup Approach," *ArXiv200604628 Cs Stat*, Jun. 2020, Accessed: Mar. 29, 2021. [Online]. Available: http://arxiv.org/abs/2006.04628

[137] L. Yang and A. Shami, "On hyperparameter optimization of machine learning algorithms: Theory and practice," *Neurocomputing*, vol. 415, pp. 295–316, Nov. 2020, doi: 10.1016/j.neucom.2020.07.061.

[138] "Open Neural Network Exchange," *GitHub*. https://github.com/onnx (accessed Mar. 09, 2021).

[139] "ONNX Runtime." https://www.onnxruntime.ai/ (accessed Apr. 10, 2021).

[140] H. Song, F. Qin, P. Martinez-Julia, L. Ciavaglia, and A. Wang, "Network Telemetry Framework," Internet Engineering Task Force, Internet-Draft draft-ietf-opsawg-ntf-07, Feb. 2021. Accessed: Apr. 13, 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-opsawg-ntf-07

[141] H. Song *et al.*, "Postcard-based On-Path Flow Data Telemetry using Packet Marking," Internet Engineering Task Force, Internet-Draft (work in progress) draft-song-ippm-postcard-based-telemetry-09, Feb. 2021. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-song-ippm-postcard-based-telemetry-09

[142] C. Rottondi and G. Verticale, "Using packet interarrival times for Internet traffic classification," in *2011 IEEE Third Latin-American Conference on Communications*, Oct. 2011, pp. 1–6. doi: 10.1109/LatinCOM.2011.6107404.

[143] O. Osanaiye, K. R. Choo, and M. Dlodlo, “Change-point cloud DDoS detection using packet inter-arrival time,” in *2016 8th Computer Science and Electronic Engineering (CEEC)*, Sep. 2016, pp. 204–209. doi: 10.1109/CEEC.2016.7835914.

[144] P. Biondi, “Scapy - Packet crafting for Python2 and Python3.” https://scapy.net/ (accessed May 13, 2021).

[145] R. R. S, R. R, M. Moharir, and S. G, “SCAPY- A powerful interactive packet manipulation program,” in *2018 International Conference on Networking, Embedded and Wireless Systems (ICNEWS)*, Dec. 2018, pp. 1–5. doi: 10.1109/ICNEWS.2018.8903954.

[146] “iPerf - The ultimate speed test tool for TCP, UDP and SCTP.” https://iperf.fr/iperf-doc.php (accessed Apr. 15, 2021).

Curriculum Vitae

Name:	Cesar Gomez
Post-secondary Education and Degrees:	BE, Electronics Engineering Saint Thomas Aquinas University Bogota, Colombia 2000 - 2004
	MESc, Telecommunications Engineering National University of Colombia Bogota, Colombia 2007-2010
	PhD, Electrical and Computer Engineering The University of Western Ontario London, Ontario, Canada 2017-2021
Honours and Awards:	Outstanding Young Professional Award IEEE London Section November 2019
	Finalist, Discover AI Challenge 2020 Microsoft – Agorize – McGill University May 2020
	Certificate of Volunteer Appreciation IEEE London Section November 2020
Related Work Experience	Network Engineer Nortel Networks 2006-2009
	Lecturer Unitec 2014-2017
	Research and Teaching Assistant The University of Western Ontario 2017-2021

Publications:

C.A. Gomez, A. Shami, and X. Wang, “Efficient Network Telemetry based on Traffic Awareness”, *IEEE Open Journal of the Communications Society,* (submitted).

C.A. Gomez, X. Wang, and A. Shami, “Federated Intelligence for Active Queue Management in Inter-Domain Congestion”, *IEEE Access*, vol. 9, pp. 10674-10685, January 2021. DOI: 10.1109/ACCESS.2021.3050174.

C.A. Gomez, X. Wang, and A. Shami, “Intelligent Active Queue Management Using Explicit Congestion Notification”, *IEEE Global Communications Conference, GLOBECOM 2019*, December 2019. DOI: 10.1109/GLOBECOM38437.2019.9013475.

C.A. Gomez, A. Shami, and X. Wang, “Machine Learning Aided Scheme for Load Balancing in Dense IoT Networks”, *Sensors*, vol. 18, no. 11, p.3779, November 2018. DOI: 10.3390/s18113779.

C.A. Gomez and J.E. Ortiz, “A Greener Method for Content Sharing in Mobile Ad Hoc Networks”, *IEEE Latin-American Conference on Communications, LATINCOM 2010,* pp. 1-6, September 2010. DOI: 10.1109/LATINCOM.2010.5640998.