Performance Evaluation And Anomaly detection in Mobile BroadBand Across Europe

by

Mohamed Lamine Touhami Moulay Brahim

in partial fulfillment of the requirements for the degree of Doctor in

Multimedia And Communication

Universidad Carlos III de Madrid

Advisor: Vincenzo Mancuso

June 2022

Performance Evaluation And Anomaly detection in Mobile BroadBand Across Europe

Prepared by:
Mohamed Lamine Touhami Moulay Brahim, IMDEA Networks Institute, Universidad Carlos III de Madrid
contact: mohamed.Moulay@imdea.org

Under the advice of:
Vincenzo Mancuso, IMDEA Networks Institute
Telematic Engineering Department, Universidad Carlos III de Madrid

This work has been supported by:

institute IMDEA networks

To my father, for making this possible and to all my family for the constant support.

To my soon to be wife, thank you for all the love and support changing the meaning of long distance.

Acknowledgements

Starting a Ph.D. is a long journey that requires patience, detection, and hard work to reach the final stretch, and I wouldn't have been able to do it without the guidance and help of my supervisor Dr. Vincenzo Mancuso. He guided me to become a better version of myself through constructive feedback and passing on responsibilities preparing me to stair the ship on my own one day. Combined with the guidance of Dr. Antonio Fernández Anta and Rafael García Leiva, I learned a lot, and I can't thank you enough. I can't also forget to thank Dr. Miguel Peón Quirós for his help in integrating me into the research team and guiding me with the tasks at hand. I would also like to extend my thank you to my colleagues at IMDEA Networks. The environment was amazing, filled with humor and good times. Since the second floor was my home during that period, I would like to thank Joan, Maurizio, Foivos, Ander, Roberto, Javier, Sonia, Pilar, Vadim, Noelia, Constantine, Yago, and Hany for those memories and moments.

Finally, I would like to thank all IMDEA Network's team from professors to HR for being there for us.

Published Content

The ideas and investigations of this Thesis emerged from the following refereed publications:

[**1**] **Mohamed Moulay**, Vincenzo Mancuso. Experimental performance evaluation of WebRTC video services over mobile networks. Published in *The 5th International Workshop on Computer and Networking Experimental Research using Testbeds in conjunction with IEEE INFOCOM 2018*, July 2018. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8407020

- This work is fully included and its content is reported in Chapter 4.
- The author implemented a Docker container that assets the performance of WebRTC in mobile-cellular networks leveraging the MONROE testbed. The author investigated Key Performance Indicators (KPIs) such as BitRate, Jitter, Frames Per Second (FPS), and Packet delay in static and mobile networking environments. At the time of writing the paper, no such work existed to evaluate WebRTC in real-world scenarios.

[**2**] Cise Midoglu, **Mohamed Moulay**, Vincenzo Mancuso, Ozgu Alay, Andra Lutu, Carsten Griwodz. Open video datasets over operational mobile networks with MONROE. Published in *The Proceedings of the 9th ACM Multimedia Systems Conference, June 2018, (MMsys '18).* https://dl.acm.org/doi/abs/10.1145/3204949.3208138

- This work is partially included and its content is reported in Chapter 4.
- The author participated in writing several parts of this paper and his role was centred around the inclusion of WebRTC in terms of data-set, configuration, descirption, and results.

[**3**] **Mohamed Moulay**, Fernando Diez, Vincenzo Mancuso. On the Experimental Assessment of QUIC and Congestion Control Schemes in Cellular Networks. Published in *19th Mediterranean Communication and Computer Networking Conference (IEEE MedComNet 2021)* , 15-17 June 2021, Online Conference. https://doi.org/10.1007/978-3-030-05195-2_20

- This work is fully included and its content is reported in Chapter 5.
- The author's initial investigation and performance evaluation of Quick UDP Internet Connections (QUIC) utilizing the MONROE testbed in different mobile network environments. The key idea was to evaluate QUIC using different congestion control algorithms as well as leveraging the IETF QUIC implementation.

[4] **Mohamed Moulay**, Rafael Garcia and Maroni, Pablo J. Rojo and Lazaro, Javier and Mancuso, Vincenzo and Anta, Antonio Fernandez. A Novel Methodology for the Automated Detection and Classification of Networking Anomalies. Published in *The 3rd International Workshop on Network Intelligence (NI 2020): Learning and Optimizing Future Networks in conjunction with IEEE INFOCOM 2020, Virtual, July 2020.* https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9162710

- This work is fully included and its content is reported in Chapter 6.
- The author participated in writing several parts of this paper and his role in this work is focused on implementing and validating the methodology using multiple classical and interpretable machine learning algorithms.

[5] **Mohamed Moulay**, Rafael Garcia, Mancuso, Vincenzo, Anta, Antonio Fernandez and Pablo J. Rojo. TTrees: Automated Classification of Causes of Network Anomalies with Little Data. Published in *The 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (IEEE WoWMoM 2021)*, 7-11 June 2021, Fully virtual event. https://eprints.networks.imdea.org/2303/1/TTreess.pdf

- This work is fully included and its content is reported in Chapter 7.
- The author participated in writing several parts of this paper and his role in this work is focused on improving the methodology and automating the anomalies detection in cellular networks.

[6] Vincenzo Mancuso and Miguel Peon Quiros and Cise Midoglu and **Mohamed Moulay** and Vincenzo Comite and Andra Lutu and Ozgu Alay and Stefan Alfredsson and Mohammad Rajiullah and Anna Brunstrom and Marco Mellia and Ali Safari Khatouni and Thomas Hirsch. Results from running an experiment as a service platform for mobile broadband networks in Europe. *Computer Communications vol 133, Pages 89-101, Elsevier.* https://eprints.networks.imdea.org/1972/1/COMCOM_5773.pdf

- This work is partially included and its content is reported in Chapter 4.
- The author participated in writing the WebRTC part of this paper and his role in this work is focused on a more broad assessment of WebRTC in cellular networks.

[7] **Mohamed Moulay** and Rafael Garcia and Vincenzo Mancuso and Pablo Rojo and Antonio Fernandez Anta and Ali Safari Khatouni. MonTrees: Automated Detection and Classification of Networking Anomalies in Cellular Networks. Under Review in *IEEE Transactions on Network and Service Management*, 2021, Virtual. https://arxiv.org/pdf/2108.13156.pdf.

- This work is partially included and its content is reported in Chapter 7.
- The author's role in this work is focused is to further evaluate Strees leveraging various data-sets with multiple file download scenarios.

[8] **Mohamed Moulay**, Rafael Garcia, Pablo J. Rojo, Fernando Diez, Mancuso, Vincenzo, Anta, Antonio Fernandez. Automated Identification of Network Anomalies and Their Causes with Interpretable Machine Learning: the CIAN Methodology and TTrees Implementation. Accepted in *The International Journal for the Computer and Telecommunications Industry Special Issue with WoWMoM.*

- This work is fully included and its content is reported in Chapter 7.
- The author participated in writing several parts of this paper and his role in this work is focused on extending the methodology with the QUIC data-set.

Abstract

With the rapidly growing market for smartphones and user's confidence for immediate access to high-quality multimedia content, the delivery of video over wireless networks has become a big challenge. It makes it challenging to accommodate end-users with flawless quality of service. The growth of the smartphone market goes hand in hand with the development of the Internet, in which current transport protocols are being re-evaluated to deal with traffic growth. QUIC and WebRTC are new and evolving standards. The latter is a unique and evolving standard explicitly developed to meet this demand and enable a high-quality experience for mobile users of real-time communication services. QUIC has been designed to reduce Web latency, integrate security features, and allow a high-quality experience for mobile users. Thus, the need to evaluate the performance of these rising protocols in a non-systematic environment is essential to understand the behavior of the network and provide the end user with a better multimedia delivery service. Since most of the work in the research community is conducted in a controlled environment, we leverage the MONROE platform to investigate the performance of QUIC and WebRTC in real cellular networks using static and mobile nodes. During this Thesis, we conduct measurements of WebRTC and QUIC while making their data-sets public to the interested experimenter. Building such data-sets is very welcomed with the research community, opening doors to applying data science to network data-sets. The development part of the experiments involves building Docker containers that act as QUIC and WebRTC clients. These containers are publicly available to be used candidly or within the MONROE platform. These key contributions span from Chapter 4 to Chapter 5 presented in Part II of the Thesis.

We exploit data collection from MONROE to apply data science over network data-sets, which will help identify networking problems shifting the Thesis focus from performance evaluation to a data science problem.

Indeed, the second part of the Thesis focuses on *interpretable* data science. Identifying network problems leveraging Machine Learning (ML) has gained much visibility in the past few years, resulting in dramatically improved cellular network services. However, critical tasks like troubleshooting cellular networks are still performed manually by experts who monitor the network around the clock.

In this context, this Thesis contributes by proposing the use of simple interpretable ML algorithms, moving away from the current trend of high-accuracy ML algorithms (e.g., deep learning) that do not allow interpretation (and hence understanding) of their outcome. We prefer having lower accuracy since we consider it interesting (anomalous) the scenarios misclassified by the ML algorithms, and we do not want to miss them by overfitting. To this aim, we present CIAN (from Causality Inference of Anomalies in Networks), a practical and interpretable ML methodology, which we implement in the form of a software tool named TTrees (from Troubleshooting Trees) and compare it to a supervised counterpart, named STress (from Supervised Trees). Both methodologies require small volumes of data and are quick at training. Our experiments using real data from operational commercial mobile networks e.g., sampled with MONROE probes, show that STrees and CIAN can automatically identify and accurately classify network anomalies—e.g., cases for which a low network performance is not justified by operational conditions—training with just a few hundreds of data samples, hence enabling precise troubleshooting actions. Most importantly, our experiments show that a fully automated unsupervised approach is viable and efficient. In Part III of the Thesis which includes Chapter 6 and 7.

In conclusion, in this Thesis, we go through a *data-driven* networking *roller coaster,* from performance evaluating upcoming network protocols in real mobile networks to building methodologies that help identify and classify the root cause of networking problems, emphasizing the fact that these methodologies are easy to implement and can be deployed in production environments.

Table of Contents

Acknowledgements vii

Published Content ix

Abstract xiii

Table of Contents xv

List of Tables xix

List of Figures xxi

List of Acronyms xxvii

I Introduction 1

1. Introduction 3

1.1. Contribution 6

1.2. Outline of the Thesis 8

2. Background 11

2.1. Mobile Broadband 11

2.2. Rising Protocols 12

2.2.1. Web Real-Time Communication 12

2.2.2. Quick UDP Internet Protocol 13

2.3. Data Science In Cellular Networks 15

2.3.1. Causality 15

2.3.2. Explainable AI 16

2.3.3. Anomaly detection 17

3. MONROE 19

3.1. EaaS platform design and implementation 21

3.1.1. Node instrumentation 23
3.1.2. Data flows 23
3.1.3. At the node side 25
3.1.4. At the server side 25
3.1.5. Access to data 26
3.1.6. User access and experiment scheduling 26
3.1.7. Experimentation workflow 27
3.2. Experiments 27
3.2.1. Experiments currently available as services 27
3.2.2. Selected measurement studies 32
3.3. Discussions 39

II Network protocols performance evaluation 41

4. Performance Evaluation of WebRTC 43
4.1. WebRTC Overview 44
4.1.1. Real-time communications to and from browsers 44
4.1.2. Protocols and Communication Services 46
4.2. Measurement Setup & data-set 47
4.2.1. Setup 47
4.2.2. data-set 48
4.3. Results and Observations 50
4.4. Discussion 55

5. Performance Evaluation of QUIC 57
5.1. QUIC operation and logging 59
5.1.1. The QUIC protocol in a nutshell 59
5.1.2. QUIC logging 59
5.2. Experimental Methodology 61
5.2.1. Methodology 63
5.2.2. Setup 65
5.3. Results 66
5.3.1. Assessment of QUIC and HTTP/3 performance 66
5.3.2. Assessment of Congestion Control Variants in QUIC 69
5.4. Discussion 72

III Data science in cellular networks 75

6. Supervised Trees 77

6.1. Supervised ML Methodology 78
6.1.1. Target Variable Characterization 81
6.1.2. Detecting Anomalies 81
6.1.3. Clustering Anomalies 82
6.1.4. Classifying Anomalies 83
6.2. data-sets 84
6.2.1. Nokia drive-test measurements 84
6.2.2. MONROE measurements 85
6.3. Results 86
6.3.1. Data Analysis for Nokia Drive Tests 86
6.3.2. Data Analysis for MONROE data-sets 94
6.4. Discussion 104

7. Causality Inference of Anomalies in Networks 105
7.1. Overview of CIAN 108
7.1.1. The Core Idea for Detecting Anomalies 109
7.1.2. Input 110
7.1.3. Discretization 111
7.1.4. Selection of Anomalous Scenarios 111
7.1.5. Selection of the Most Relevant Features 112
7.1.6. Clustering Using the Most Relevant Features 113
7.1.7. Aspect Classification of Scenarios 114
7.1.8. Using the Classifiers to Detect and Identify Anomalies 114
7.2. Implementation of the CIAN methodology 115
7.2.1. Data Preparation in TTrees 116
7.2.2. Discretization in TTrees 116
7.2.3. Knowledge Model for Identifying Anomalies 117
7.2.4. Most Relevant TTrees Features 118
7.2.5. Clustering of Anomalies in TTrees 120
7.2.6. Aspect Classification in TTrees 121
7.2.7. Software Implementation 122
7.3. Empirical Evaluation 122
7.3.1. Supervised ML-based Troubleshooting 123
7.3.2. Datasets 123
7.3.3. TTrees in Action and its Validation 126
7.3.4. TCP Performance Evaluation 133
7.3.5. QUIC performance evaluation 136
7.3.6. Modifications needed to deal with imbalancedness of data-sets . . 140
7.4. Discussion 149

8. Conclusions **151**

References **153**

List of Tables

3.1. MONROE metadata topics 29

4.1. MONROE nodes throughput comparison 48

6.1. Overview of MONROE data-sets collected in Norway, Sweden, Italy, and Spain 85

6.2. Summary of the decision tree rules of Figure 6.4 producing the most significant RTT attributes against the TDR percentile class split shown in Figure 6.3 88

6.3. Confusion matrix for the supervised ML classifier of Figure 6.4 trained with the classes identified using Figure 6.3 89

6.4. radio and TCP data attributes used by the unsupervised ML algorithm used in STrees (i.e., k-means, with $k = 2$) 90

6.5. Highlights of the STrees decision tree rules for detection of anomalies in Figure 6.8 showcasing the dominant attributes from Table 6.4 and classes from Figure 6.8 . 93

6.6. A summary of the initial decision tree's performance utilizing Facebook as a service in all the countries part of the MONROE project with all their operators 96

6.7. The final decision tree performance with STrees, using MONROE data-sets for Facebook and Google experiments. 99

6.8. A summary of the initial decision tree's performance utilizing Google as a service following up the same methodology steps from section 6.3.2.1 . . . 100

6.9. The final STrees decision tree performance with MONROE data-sets for YouTube and Twitter experiments. 103

7.1. Basic notation. 109

7.2. Brief description of the experimental data-sets used in this work 123

7.3. Knowledge (C_1) and Aspect Classification (C_2) trees' performance across the different data-sets using multiple aspect selection algorithms (Mutual Information, Joint Mutual Information, and Miscoding) 134

7.4. Classification output of TTrees with QUIC experiments – Data grouped per congestion control algorithm (classes with no samples are omitted) . . 135

7.5. Classification output of TTrees with QUIC experiments – Data grouped per experiment execution type (classes with no samples are omitted) . . . 135

7.6. A summary of the knowledge tree's (C_1) performance in TTrees utilizing Facebook & Google as a service in all the countries part of the MONROE project with all their operators. 141

7.7. A summary of the aspect classification tree's (C_2) performance with TTrees utilizing Facebook & Google as a service in all the countries part of the MONROE project with all their operators. 142

7.8. A comparison of the number of aspects and aspect families extracted from C_2 between k-means and GMM for the Google & Facebook data-sets in all the countries part of the MONROE project. The table also shows the number of aspects that are common to both algorithms as well as the ratio, calculated from the division of the intersection by the size of the smallest set of aspects between k-means and GMM. 147

List of Figures

1.1. High-level illustration of the Thesis. 8

2.1. QUIC communication flow. 14

3.1. MONROE system design. Researchers access the system through the Web user interface and scheduler, or directly through the various repositories and data bases. Several passive (metadata, mPlane, etc.) and active (RTT, HTTP bandwidth, etc.) probes monitor network usage and performance continuously. 22

3.2. Flow of information in the MONROE platform. 24

3.3. Experiment workflow covering the design, test and experimentation phases. 27

3.4. Experiments currently available as services that can be run on the MONROE platform, within an EaaS framework. 28

3.5. RTT and RSSI measured in a bus at Karlstad, Sweden, over a few observation days. Average RSSI values are shown on the XY plane. Individual RTT measures are plotted on the Z-axis using their relative timestamps as height to visualize successive laps. 33

3.6. This representation of link technology for the bus at Karlstad reveals that 4G coverage is consistently available for the complete route during the analyzed period. 33

3.7. TCP three-way handshake times (TWHT) obtained using the HTTP download experiment for bandwidth measurement with different operators versus the RSSI reported in MONROE metadata. Blue and red correspond to 4G and 3G samples, respectively. 34

3.8. Violin plots of the RTT measurements for different operators in Spain (ES), Norway (NO) and Sweden (SE). 35

3.9. MONROE-Nettest base experiment results. 36

3.10. Average Time to First Byte and Complete Page Load Time for some operators in Spain (ES), Norway (NO) and Sweden (SE) for www.bbc.com. 37

3.11. Country-wise per-operator overall webpage download performance. 38

4.1. WebRTC peer-to-peer communication. 45
4.2. MONROE-WebRTC video streaming setup. 47
4.3. WebRTC performance experienced by static nodes under different operators in different countries. 51
4.4. WebRTC performance figures observed for static nodes. 52
4.5. WebRTC performance figures observed for a bus in a medium-size city in Sweden. 53
4.6. WebRTC video performance measured at destination, on a public bus on service in a medium-size city in Sweden. 54
4.7. WebRTC performance figures observed for a train in a medium-size city in Norway. 56

5.2. QUIC events used in *qlog* 61
5.3. Example of `qvis` statistics view 62
5.4. Experimental scenario 64
5.5. `flowsim` MONROE platform setup 64
5.6. Download time comparison between QUIC and TCP, with `flowsim` clients in Spain, Sweden and Norway (download size of 1 MB) 67
5.7. Download time comparison of QUIC and TCP with `flowsim` on a public bus in Sweden (1 MB downloads) 67
5.8. Page download time comparison between HTTP versions with `flowsim` clients in Spain (static) and Sweden (mobile), with page size of 100 kB . . 68
5.9. Time series for for a QUIC download from a mobile `flowsim` client in Sweden (*qlog* events, 100 kB downloads) 68
5.10. Time series for a webpage download from a mobile `flowsim` client in Sweden (*qlog* events, 100 kB downloads) 69
5.11. Sequential time series in Spain with three download streams (*qlog* files of `Mvfst` experiments) 70
5.12. Parallel time series in Spain with three download streams (*qlog* files of `Mvfst` experiments) 70
5.13. Parallel time series in Norway (with a static `Mvfst` client) with three download streams 71
5.14. Sequential time series in Sweden (with a mobile `Mvfst` client) with three download streams 72
5.15. Parallel time series in Sweden (with a mobile `Mvfst` client) with three download streams 73

6.1. High-level presentation of the STrees methodology, starting with the characterization of the target variable, followed by the initial classification of samples using the RTT as the data attribute of choice, grouping anomalies with *k*-means (with Radio and TCP related data attributes), and final classification of anomalies with a decision tree. 80

6.2. Possible causes of a performance anomaly, starting with the RTT as the first choice and moving downwards to identify other factors that cause anomalies in the TDR observed (when using TCP for downloading files). The conclusion may be that the cause of the observed anomalies is not identifiable given the current model. 81
6.3. Distribution of Throughput Data Rate. 86
6.4. Decision tree generated with supervised ML using RTT attributes as input and TDR classes inferred from percentiles (see Figure 6.3). when visualizing the tree each box has the quality of the gini split, the number samples at each split, the number values for each TDR class (Bad, OK, Good), and which class was picked. 87
6.5. Graphical plot of the data items properly classified by the decision tree of Figure 6.4 in black, and the misclassified data items in red. 89
6.6. The two k-means clusters obtained with TCP data attributes with respect to the congestion window average and maximum value attributes. 91
6.7. The two k-means clusters obtained with radio attributes with respect to the RSSI and RSRP attributes. 91
6.8. STrees anomaly detection decision tree with TCP, radio attributes from Table 6.4, and k-means labeled clusters as classes. when visualizing the tree each box has the quality of the gini split, the number samples at each split, the number values for each class (Failure to identify, Radio Ok/TCP Problem, Radio Problem/TCP Ok, unknown), and which class was picked during each split. 92
6.9. The outcome of the misclassified points using a combination of unsupervised and supervised ML in Figure 6.8 and properly classified points with the supervised ML classifier in Figure 6.4. 93
6.10. A snapshot of the Distribution of Throughput Data Rate in Sweden using Operator 0 and Facebook as a service. 95
6.11. The depth versus accuracy for a decision tree with the confidence interval using Facebook as service in Sweden with Operator 0. 95
6.12. The two k-means clusters per countries and operators obtained with radio attributes concerning the RSSI and RSRP attributes using anomalies sample from table 6.6 leveraging Facebook as a service. 97
6.13. The two k-means clusters per countries and operators obtained with TCP attributes using anomalies sample from table 6.6 leveraging Facebook as a service 98
6.14. A snapshot of the Distribution of Throughput Data Rate in Sweden using Operator 0 and Google as a service. 100

6.15. The two k-means clusters per countries and operators obtained with radio attributes concerning the RSSI and RSRP attributes using anomalies sample from table 6.8 leveraging google as a service. 101

7.1. Steps of TTrees (using multiple ML techniques): beginning with data preparation, proportional discretization, training of knowledge tree, selection of the most relevant features, identification of anomaly clusters, and training of a network aspect anomaly classifier. 115

7.2. Target KPI (throughput) distribution for Dataset #1 via the proportional discretization approach 127

7.3. Zoom into a subset of the knowledge tree built for Dataset #1. The figure also reports which family of features the branching variable belongs to (see Table 7.2). Observe that both branches are the same up to the fourth node, after which the left corresponds to the *true* brach and the right to the *false* branch. 127

7.4. Difference (predicted - discretized) between the class assigned by the knowledge tree and the discretized category for the target KPI of Dataset #1 128

7.5. Ranking of features according to their relevance to the description of anomalies for Dataset #1 128

7.6. NID heatmap matrix for cluster pairs. Darker colors represent redundancy. 129

7.7. Network aspect selection based on miscoding; lower inertia of k-means clusters is preferred, unless clusters are redundant or imbalanced 130

7.8. Aspects obtained with TTrees for Dataset #1 (using Mscd, see Figs. 7.5 and 7.7). The parameters used are Signal-to-interference-plus-noise ratio (SINR), Number of packet acknowledgment sent, absolute TCP Window ratio, and RTT. 131

7.9. Aspects obtained with STree for Dataset #1 132

7.10. A comparison between the supervised and unsupervised approach showing the resulting aspect classifiers 133

7.11. Aspects obtained with TTrees for Dataset #4 136

7.12. Zoom into a subset of the branches of the knowledge tree C_1 built by TTrees for Dataset #4. The figure also reports which family of features the branching variable belongs to (see Table 7.2) 137

7.13. Zoom into a subset of the branches of the aspect classification tree C_2 built by TTrees for Dataset #4. The figure also reports which family of features the branching variable belongs to (see Table 7.2) 138

7.14. Histogram of Inertia (a, b) and BIC (c, d) values extracted during aspect clustering for all relevant aspects and with two operators in Spain and Italy. The lower the BIC or inertia the cleaner cluster grouping and splitting of samples is overall 145

7.16. Cluster grouping using *k*-means and GMM as the cluster algorithm using Google Dataset and *op*0_*it* 148

7.17. Cluster grouping Comparsion using *k*-means and GMM as the cluster algorithm for Dataset #1 148

7.15. Histogram of the ratio of samples falling under the non-problematic class during aspect clustering for all relevant aspects in Dataset #5 and with two operators in Sweden and Norway. As it can be seen, most aspects, especially with *k*-means, yield imbalanced clusters of anomalies. 149

List of Acronyms

API Application Programming Interface

BI Byte Index

CLI Command Line tool

CART Classification And Regression Tree algorithm

CIAN Causality Inference of Anomalies in Networks

DB Database

DASH Dynamic Adaptive Streaming over HTTP

DSCP Differentiated Services Code Point

EaaS Experiment as a Service

ECDF Empirical Cumulative Distribution Function

EDGE Enhanced Data for Global Evolution

FPS Frames Per Second

GPS Global Positioning System

GMM Gaussian Mixture Model

GSM Global System for Mobile Communications

GIPS Global IP Solutions

GDPR General Data Protection Regulation

HAR HTTP Archive

HoL Head-of-Line

HTTP Hypertext Transfer Protocol

ICE Interactive Connectivity Establishment

JMI Joint Mutual Information

JSEP JavaScript Session Establishment Protocol

KPI Key Performance Indicator

LTE Long Term Evaluation

MI Mutual Information

ML Machine Learning

MBB Mobile BroadBand

mscd Miscoding

OI Object Index

OS Operating System

PCA Principal Component Analysis

PLT Page Load Time

QoE Quality of Experience

QoS Quality of Service

QUIC Quick UDP Internet Connections

RTC Real Time communication

RTT Round Trip Time

RMBT RTR Multithreaded Broadband Test

RMSE Root Mean Square Error

RSSI Received Signal Strength Indicator

RTTCC Receive-side real-time congestion control

SDP Session Description Protocol

SNR Signal-to-Noise Ratio

SCTP Stream Control Transport Protocol

SRTP Secure Real-Time Transport

STress Supervised Trees

TR Test Record

TCP Transmission Control Protocol

TDR Throughput Data Rate

TTrees Troubleshooting Trees

UMTS Universal Mobile Telecommunications System

WebRTC Web Real-Time Communication

Part I

Introduction & Background

The rise of the Internet to be the more sophisticated tool ever created by humans, made our lives easier, allowing us to be informed, stream our favorite shows, keep in touch with family and friends, and express our ideas to large audiences over social media networks. The most popular device to do so is our smartphone, which enables instant Internet access anywhere and anytime. Therefore, Mobile BroadBand (MBB) networks have become a crucial infrastructure for people to stay connected everywhere and while on the move. Society's increased dependence on MBB networks motivates researchers and engineers to enhance the capabilities of mobile networks by designing new technologies and protocols to cater to plenty of new applications and services that require new monitoring tools and data analytics. Thus, Part I of this Thesis provides a background of the protocols and methodologies dealt with during the Thesis. Chapter 1 describes a comprehensive introduction focusing on MBB, Internet traffic measurements, and the vast potential of collecting data and understanding network protocols. It also describes the key contributions of the Thesis. Chapters 2 represents the related work and technical background. Finally, Chapter 3 introduces the testbed platform used in the Thesis.

1 Introduction

As everyone knows, people are more and more connected to the Internet, with mobile terminals allowing access to information from anywhere and anytime. The evolution of the cellular network is something to admire, from the first cellular network named Global System for Mobile Communications (GSM) to 5G moving through Enhanced Data for Global Evolution (EDGE), Universal Mobile Telecommunications System (UMTS), and Long Term Evolution (LTE) wherein each generation we saw increased throughput and reduced latency helping move Internet access from computers connected to wired cables into smartphone-connected through cellular networks. In addition, this helped create new kinds of business such as online shopping and even new career opportunities like Youtube content creators and social influencers, changing the way we see the Internet into a golden opportunity. Considering this growth, many researchers are focusing on understand the behavior of Internet so as to be able to evaluate and predict the performance of different technologies, the user' Quality of Experience (QoE) and the Quality of Service (QoS). Given the large quantity of data to gather and parse, this effort entails developing automatic, reliable and efficient measurement probes and data analytic tools. However, some key aspects such as looking at ways to troubleshoot cellular networks remain a substantially manual procedure. Indeed, highly skilled experts analyze alarms and statistics of performance indicators regularly to detect and diagnose the cause of problems in the network. In contrast, unskilled engineers can not even detect problems effectively [9].

In this context, there is a need to explore and evaluate the behavior of new protocols, such as Web Real-Time Communication (WebRTC) and Quick UDP Internet Connection (QUIC).

Web Real-Time Communication proposes to easily integrate video services in Web browsers, based on local tools [10]. In turn, such tools are based on well-known Web technologies, able to integrate audio, video, and data transfer operations of the real-time communication protocol (RTC) into a standard webpage. The WebRTC project[1] was

[1] http://www.webrtc.org/

first introduced by Google as an open-source project. Then other software developers and telecom vendors joined, which has led to the integration of WebRTC into commercial browsers like Chrome, Opera, and Firefox [11, 12].

QUIC is a new network transport protocol initially designed and proposed by Google, which operates on top of a more straightforward transport protocol, UDP, to bypass legacy TCP/IP transport limitations. QUIC itself is commonly seen as a transport-layer protocol, although it embeds encryption features and relies on conventional UDP transport.

QUIC eliminates significant TCP bottlenecks such as the need for an initial TCP handshake mechanism, which takes one Round Time Trip (RTT) or two, in the case of TCP data encryption via TLS. Indeed, Google has designed and implemented a novel encryption scheme for QUIC, similar to TLS, which couples connection establishment and key agreement within one RTT. Nevertheless, QUIC can start a connection in zero RTT, like UDP, by instantly sending encrypted application data to the server. This is possible when it previously has cached in a server certificate from a previous connection. Moreover, running on top of UDP, QUIC bypasses the Head-of-Line (HoL) issue of TCP in case of packet loss. For these reasons, in the long run, QUIC is expected to replace TCP and TLS in the Web [13].

While researchers have carried performance evaluations of WebRTC and QUIC in different networking scenarios, the problem is that most of them were under simulated or controlled environments. For instance, the authors of [14] used a cloud-engineered automatic testing tool for WebRTC, although they have not tested the service offered by mobile operators and core networks.

Similarly, the authors of [15] have experimentally tested one-to-many communications over WebRTC (namely "simulcast"), although their experiments are limited to a gigabit LAN environment.

For what concerns QUIC, the authors of [16] use Google's server and the client to evaluate the performance of QUIC and TCP in wireless networks in a controlled environment. However, they have not tested the service offered by mobile operators and core networks. Similarly, the authors of [17] have QUIC and TCP in the kernel level through a gigabit LAN environment.

Evaluating and assessing new protocols is one part of the story. To improve the overall QoS and QoE, there is a need to investigate on automating the detection of networking problems using new technologies, and in particular by leveraging the raising paradigm of ML. This is made possible by applying data science to data collected from simulating those protocols or through measurement campaigns. To deal with these needs, operators are investing resources into the automation of the maintenance and troubleshooting tasks through self-healing functionalities within the scope of intelligent, self-organizing network operation tools. Self-healing networks are responsible for detecting, identifying, and making decisions on recovery actions [18]. Multiple proposals exist for making fault detection and self-healing systems practical in mobile networks [19]. However, current self-healing troubleshooting proposals lack flexibility and do not scale well. There are newly-defined approaches based on ML, which use deep learning and neural networks (as black boxes) [20]. Unfortunately, these approaches lack interpretability, so it is not possible to understand the cause of a detected network problem, and manual intervention is required for classification after the detection. It is then possible to resort to Explainable AI, or XAI, which deals with the problem of how human users could understand AI's cognition and decide if an AI-based model can be trusted or not [21]. However, XAI does not help interpret AI decisions and conclusions *per se*. Multiple methods have been proposed to address the complex issue of ML interpretability. [22], from determining which features contribute the most to a neural network's output to the development of targeted models that explain individual predictions. Nonetheless, as of today, troubleshooting cellular networks is a manual task.

Driven by all the above observations, this Thesis develops different ideas and solutions to solve real data collection for new protocols and automatic analysis, with automatic identification of the causes of anomalous behaviors. We leverage the MONROE platform to probe real commercial MBBs.

We start by evaluating WebRTC video services performance linked to using an off-the-shelf WebRTC-based streaming application with MBB networks in mobile and stationary scenarios. Then, we collect data from real-world scenarios from multiple European countries and make them public by using open-science tools such as Zenodo. Alongside, we develop an application that enables streaming videos in real-time with high quality using Web browsers that support WebRTC and look at stats collected from the WebRTC internal page found nearly in all modern Web browsers.

Next, we look at QUIC performance figures in MBB following the same measurement methodology as WebRTC. However, the difference is that we look at various congestion control algorithms and work with multiple open-source implementations. We also resource to *qlog files*, to extract statistics [23].

With the data-sets collected from MONROE through measurement campaigns and with more data obtained by Nokia, we next build a methodology that automates the classification of networking anomalies by using interpretable ML algorithms such as decision trees and k-means. We validate the proposed methodology using different file download scenarios and popular Web service pages. We show the ease of implementing such tools in real-world production environments. We finally introduce new tweaks to the methodology to eliminate hyper-parameters. More specifically, we propose a new feature selection algorithm and mixture of supervised, unsupervised ML algorithms in tandem. Understanding and classifying these anomalous scenarios allow alerting the appropriate department to take corrective actions.

In the following subsection, we detail the contribution of this Thesis.

1.1. Contribution

This section presents a quick recap covering the significant Thesis contributions mapped to the associated research question. The major contributions are summarized into 4 points and are as follow:

Contribution 1. *Performance Evaluation of WebRTC Video Services Over Mobile Networks*

Given that WebRTC was pretty new around the investigation time, most of the performance evaluation experiments were over-controlled or simulated environments. There was a need to fill a gap in a WebRTC video service measurement campaign over mobile networks. Thus, we developed and carried out a measurement campaign found in Chapter 4 of WebRTC leveraging MONROE testbed flexibility testing WebRTC over different European countries and in different MBB scenarios. We benchmarked WebRTC by collecting stats regarding video quality such as bitrate, frames per second (FPS), jitter and packet loss. Later on, this came in handy to explain WebRTC performance degradation and challenges encountered in production environments.

Contribution 2. *On the Experimental Assessment of QUIC and Congestion Control Schemes in Cellular Networks*

This contribution highlights QUIC's and HTTP3 performance in mobile networks. Experiments are performed with the help of the MONROE testbed. A blend of stats like the file download time, round trip time (RTT), bytes in flights, throughput, congestion window encapsulated in qlog file while leveraging two QUIC open-source implementations. The interaction of QUIC transport with the choice of the congestion control algorithm is investigated. Furthermore, the performance figures of different congestion control algorithms in different networking scenarios are evaluated. Finally, we conclude by observing that QUIC is advantageous over TCP for what concerns HTTP applications. In addition, we find that the congestion control algorithm, especially under mobility, strongly impacts the QUIC's overall performance and BBR yields more stable performance figures to mobile users.

Contribution 3. *A Novel Methodology for the Automated Detection and Classification of Networking Anomalies.*
The methodology offers a way of detecting cellular networking anomalies using interpretable ML algorithms and using data-sets collected in real cellular network

environments. The approach described in this Thesis is interpretable in the sense that is accessible to understand by humans using decision trees and k-means. Contrary to popular approaches, we look at misclassified samples as anomalies. Instead of discarding them, we cluster those anomaly samples and reclassify them to detect the networking problems. Furthermore, we validate our approach with data collected through a measurement campaign of different Web services, such as Facebook, Youtube, Twitter, and Google. Notwithstanding that the data-sets were obtained under very heterogeneous conditions and from very different networks, we show that we can quickly identify behavioral anomalies and that we are also able to investigate further and identify the their root causes.

Contribution 4. A Hyper-Parameter free Methodology for the Automated Detection of Networking Anomalies

As the title tells, we extend the anomaly detection methodology by tarrying away from hyper-parameters. To name a few extensions, we change how the target variable is treated using the proportional discretization approach and automate how the most relevant indicators are selected from the anomaly samples using a new feature selection algorithm named Miscoding (mscd). We test mscd performance against popular feature selection algorithms such as Mutual Information (MI) and Joint Mutual Information (JMI). Furthermore, we validate our approach with different data-sets obtained through measurement campaigns. The final outcome of our proposed methodology is a set of easy-to-interpret classification rules that automatically alert the appropriate departments for corrective actions in case of performance anomaly. The methodology only needs small volumes of samples for the performance indicators and allows automatizing network troubleshooting with high accuracy and fast training.

1.2. Outline of the Thesis

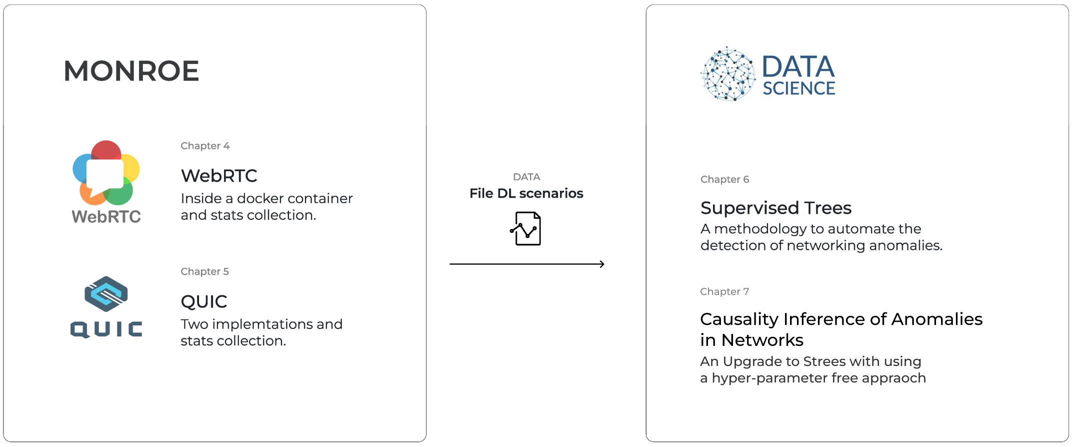

MONROE
WebRTC
Chapter 4
WebRTC
Inside a docker container
and stats collection.
QUIC
Chapter 5
QUIC
Two implemtations and
stats collection.
DATA
File DL scenarios
DATA
SCIENCE
Chapter 6
Supervised Trees
A methodology to automate the
detection of networking anomalies.
Chapter 7
Causality Inference of Anomalies
in Networks
An Upgrade to Strees with using
a hyper-parameter free appraoch

Figure 1.1: High-level illustration of the Thesis.

The outline of this Thesis is divided into eight Chapters and three Parts each Chapter showcasing a detailed contribution from Internet measurement and performance evaluation to utilizing that data to come up with methodologies for the automation and detection of cellular networking anomalies.

Part I which includes Chapter 2 and 3 focus on the related work and background of the Thesis regarding MONROE and MBB in general.

The key contribution Chapters and Parts of the Thesis are as shown in Figure 1.1.

Thus, Our first stop is the performance evaluation of WebRTC in cellular networks found in Chapter 4. Here, we develop a Docker container to evaluate WebRTC in the wild within different real networking scenarios while collecting KPIs that are deemed relevant to Video streaming—paving the way for a public repository of open data [2] collected from WebRTC experiments. Next, Chapter 5 investigates another upcoming transport protocol named QUIC. This Chapter aims to evaluate QUIC in different networking scenarios and with different congestion control algorithms. An enhancement to the public data repository from Chapter 4 is the addition of the collected data[3] from QUIC available completely for free and they represent Part II of the Thesis.

Chapters 6 and 7 presented in Part III look at cellular networks from a data science perspective. Thus, Chapter 6 is centered around building a methodology for the automated detection and classification of networking anomalies using interpretable ML algorithms and data from real cellular networks. Chapter 7 comes in as enrichment to that methodology while discarding the use of hyper-parameters.

[2] http://doi.org/10.5281/zenodo.1188411
[3] http://doi.org/10.5281/zenodo.4602217

Last but not least, Chapter 8 on its own concludes this research work.

2 Background

In this Chapter, we present the related work concerning the Thesis. We start first by describing works regarding MBB and the motivation behind the creation of MONROE. Next, we look at related works from WebRTC and QUIC. Finally, we conclude with research topics from the data science point of view.

2.1. Mobile Broadband

Due to growing interest by regulators, policy makers and networking community, several nationwide efforts to measure the performance of home and mobile broadband networks (e.g., the US FCC's Measuring Broadband America initiative [24]) have been initiated. In this Thesis we rely on the MONROE platform, which goes beyond by proposing a trans-national platform dedicated to systematic measurements. MONROE will be presented in detail in the next Chapter. Here, we point at alternative approaches to MBB performance evaluation and monitoring.

In contrast with operator-driven measurement campaigns [25–27], or existing small-case drive-by tests [28], MONROE offers open access to cross-operator collected data, including device-level metadata, which is key to interpret measurement results, across a wide variety of locations.

Moreover, there have been several crowdsourcing projects devoted to measure MBBs using tools such as MobiPerf,[1] Netalyzer [29] and Haystack [30]. Such projects allow crawling through mobile network performance factors to identify the causes of experienced performance figures. In general, such approaches lack rich metadata due to the privacy concerns created by the involvement of real users, hindering the analysis of their datasets. Also, reliance on users can provide high coverage, but at the cost of repeatability regarding location, route or equipment. However, in combination with a platform like MONROE, they could be used in a more systematic and controllable way, as proposed

[1] MobiPerf is an open source application for measuring network performance on mobile platforms: https://sites.google.com/site/mobiperfdev/

and discussed in [31]. For our work, we leverage a large-scale measurement platform and focus on users connectivity through MBB networks only. Precisely, we use the MONROE platform, which has been designed with our contribution [6]. MONROE's main goal is to allow the collection of telemetries and the execution of experiments on operative mobile networks. This is made possible through the use of hardware nodes which are connected to different broadband operators and are locally running experiments uploaded by users and programmed by means of an automatic scheduler. MONROE is currently operating as an international alliance, formed after the consortium of a previous European project. MONROE provides multi-homed, autonomous, large-scale monitoring and evaluation of performance for mobile broadband networks in heterogeneous environments. Acquiring access to this platform allows for the deployment of vast measurement setups to collect data from operational MBB networks in various European countries. Differently from other approaches based on operator-driven quality-assessment campaigns [26, 27], or on traditional drive-by tests [28], MONROE offers an open platform for repeatable and traceable experiments. Besides, it offers open access to collected data, which refer to multiple operators, and includes device-level metadata, which is the key to use and possibly filter results without raising user's privacy concerns. This offers much richer data than what can be offered by crowdsourcing initiatives like, e.g., Netalyzer [29] and Haystack [30].

2.2. Rising Protocols

In Part II of this Thesis, the focus will be on measuring the performance of two new communication protocols: WebRTC and QUIC.

2.2.1. Web Real-Time Communication

WebRTC was first developed by Global IP Solutions (GIPS) and later acquired by Google in 2011. Google's acquisition of GIPS helped make the WebRTC source code open for developers, thus beginning the standardization of WebRTC in W3C. The idea of making the WebRTC source code publicly available motivated researchers to evaluate its overall performance.

The authors of [32] introduced WebRTCBench, which measures the overall peer-to-peer communication performance. They made it publicly available, helping identify performance bottlenecks of different WebRTC implementations across a domain of platforms. The authors of [33] carried test evaluations of WebRTC applications on different browsers ad cellular networks such as 3G, 4G, and Wifi. Concluding that the overall performance in static cellular networks has a packet loss of less than 1% and advises that a round trip time of less than 100ms helps maintain a better peer-to-peer communication link. In [34], the authors evaluate the performance of various network topologies implementing WebRTC, acknowledging the congestion control techniques used and deployed lately. Receive-side real-time congestion control (RRTCC) is the algorithm employed here. Varying throughput, delay, and effects of fluctuating proportions of cross-traffic on both RTP and TCP are used to assess the performance, suggesting that RRTCC yields good results but weakens when compared with TCP. Their experimental observations showed that RRTCC works fine with low delay networks and can withstand short-term modifications that might comprise delay or queuing. The authors of [35] compares WebRTC servers on virtual machines and Docker containers. To facilitate the encoding and decoding between different browsers, the authors used the Kurento media server while checking the virtualization type that fits a WebRTC application. They carried multimedia tests on Docker containers and KVM machines, showcasing that the latter have overhead in their performance and can be expensive. Concluding that Docker containers perform better than VM's.

While current approaches evaluate WebRTC's different components. In our case, we look at WebRTC performance figures in heterogeneous networks and for users on the move leveraging the MONROE platform.

2.2.2. Quick UDP Internet Protocol

QUIC was first introduced as gQUIC (Google's version of QUIC), which led to an early investigation work [36–41] assisting the performance of that specific implementation, mostly using Chromium as the open-source gQUIC server of choice in local environments. This means that the experimenters control various dimensions that can impact the overall performance, such as network conditions. Instead, we study QUIC in the wild of operational cellular networks in different countries. We can configure QUIC servers and client, but we cannot touch the network. Figure 2.1 describes the QUIC's communication flow in terms of handshake and security.

Existing studies, e.g., [13, 42], show a toss-up in terms of who is the better protocol QUIC or TCP. They dig deep into QUIC's features, such as 0-RTT connection establishment, congestion control, and removal of TCP's HoL block. Some of these studies conclude that QUIC outperforms TCP, while others show the opposite. These conflicts between results are due to the different tuning of application settings and machine kernel used in the experiments.

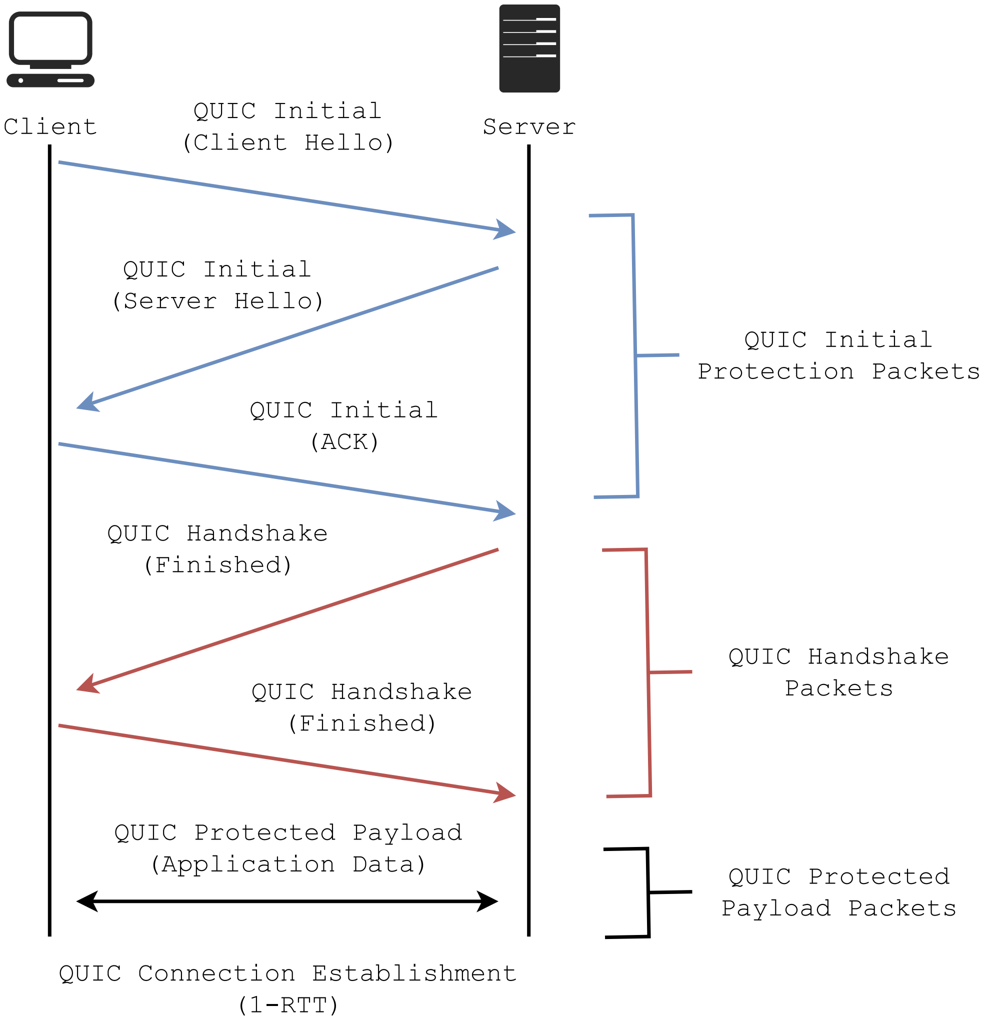

Client
Server
QUIC Initial (Client Hello)
QUIC Initial (Server Hello)
QUIC Initial (ACK)
QUIC Handshake (Finished)
QUIC Handshake (Finished)
QUIC Protected Payload (Application Data)
QUIC Initial Protection Packets
QUIC Handshake Packets
QUIC Protected Payload Packets
QUIC Connection Establishment (1-RTT)

Figure 2.1: QUIC communication flow.

Accordingly, recent works like [43] suggest to separate the QUIC protocol from the implementation since there exist multiple implementations produced by Google, Facebook, and Cloudflare. Moreover, many authors focus mostly on the gQUIC implementation of QUIC,[2] which is Google's implementation, and which differs from the proposed IETF version of QUIC,[3] as discussed by the authors of [43]. They sustain that most of the observed performance differences can be attributed to developer design and operator configuration choices when selecting the congestion control algorithm used in QUIC. They also point out that conducting new, experimental research from different implementation angles in production environments is needed. This is exactly what we present in this Thesis, leveraging two IETF-compliant open implementations of QUIC, namely `flowsim` and `Mvfst`, through which we are able to explore the impact of QUIC's

[2] https://www.chromium.org/quic
[3] https://quicwg.org

configuration parameters in real cellular networks.

Thanks to MONROE, and unlike current approaches to QUIC measurements, we focus more on the performance assessment of QUIC under real cellular networks in different European countries, showing the impact of mobility on performance figures. We believe that showcasing how QUIC behaves under operational setting, and compared to TCP alongside different congestion control algorithms, offers a broader and more realistic image of QUIC than available studies, at least for what concerns the use of QUIC in mobile networks.

2.3. Data Science In Cellular Networks

In Part III of this Thesis, the focus will shift to data science for the identification of performance anomalies.

2.3.1. Causality

Causality refers to the relationship existing between two types of events, cause and effect, where the occurrence of the latter is a consequence of the occurrence of the first. Causality must not be confused with another common term belonging to the statistics lingo, which is correlation. While the latter refers to the relationship between two variables based on their covariance (if a variable changes, the other does so too), causality establishes a clear dependency and a mapping in the way these changes take place. Causality always implies correlation but not the other way around [44].

Traditional methods for the discovery of causal relationships include the conduction of randomized trials with the objective of removing confounders, that is, variables that may deceive into drawing non-existent associations between different events. Due to the high cost of implementing these experiments, many researches have switched to the discovery of causal relationships through the exploratory analysis of observational data. This is known as causal inference [45]. As a consequence, and due to the exponential increase in the amount of data that can be collected and analyzed by systems nowadays, machine learning has been put at the forefront as a promising pathway to the automatization of causal inference and its execution in a computationally feasible amount of time. However, the introduction of machine learning models can lead to the so-called "black box effect," as the convoluted nature of these algorithms can obscure the understanding of the reasons behind the cause-effect connections and predictions discovered [46].

Advances have been done theoretically to favour the explanation of these causal relationships in the field of machine learning. Judea Pearl proposed a three-level hierarchy to classify causal information according to the type of queries each family can give answer to [47]. Firstly, the Association level, at the bottom of the causal hierarchy and syntactically expressed through conditional probability sentences, deals with information that allows to infer data associations extracted from the pure observation of data and the application of standard probabilities. Secondly, the Intervention level, regards questions that not only imply the observation of data but also the alteration of the perceived reality. The probabilistic expressions at this level can be estimated both through experiments and analysis using causal Bayesian networks. Finally, at the top of the hierarchy we have the Counterfactual level, which deals with retrospective reasoning and the hypothesis of scenarios provided environmental conditions had been different or altered prior to observation.

Pearl's three-tier hierarchy sheds light on why most machine learning systems, which are only capable of extracting data associations and thus fit in the bottom of the causal hierarchy, are unable to reason about newly unobserved data and provide causal explanations. Thus, this raises the question whether artificial intelligence can be enriched to provide new layers of causal inference. The main pathway towards a second generation of machine learning models capable of achieving this level of cognition involves the development of Explainable AI (XAI).

2.3.2. Explainable AI

Using Machine learning to detect anomalies has been around for a while. Currently, the most powerful algorithms are based on Neural Networks, which show high accuracy but have little interpretability. In reality, interpretability mainly refers to the intuition behind the outputs of a model; the more interpretable a machine learning system is, the easier it is to identify causality within the system's inputs and outputs. Thus, XAI has gained much momentum in the past few years to help explain any black-box Model. The most complete techniques to achieve that are the local interpretable model-agnostic explanations (LIME) and Shapley Additive explanations (SHAP), based on the current state of the art.

The LIME [48] method is one of the most popular interpretability methods for black-box models. Following a simple yet powerful approach, LIME can generate interpretations for single prediction scores produced by any classifier. For any given instance and its corresponding prediction, simulated randomly-sampled data around the neighborhood of the input instance for which the prediction was produced are generated. Subsequently, new predictions are made for the generated instances and weighted by their proximity to the input instance while using the model in question. Lastly, a simple, interpretable model, such as a decision tree, is trained on this newly-created dataset of perturbed instances. By interpreting this local model, the initial black-box model is consequently interpreted. Although LIME is powerful and straightforward, it has its drawbacks. In 2020, the first theoretical analysis of LIME [48] was published, validating the significance and meaningfulness of LIME and proving that poor parameters choices could lead LIME to miss out on essential features.

SHAP [49] is a game-theory-inspired method that attempts to enhance interpretability by computing the essential values for each feature for individual predictions. Firstly, the authors define the class of additive feature attribution methods, which unifies six current methods, including LIME [48], DeepLIFT [50], and Layer-Wise Relevance Propagation [51], which all use the same explanation model. Subsequently, they propose SHAP values as a suitable feature importance measure that maintains three desirable properties: local accuracy, missingness, and consistency. Finally, they present several different methods for SHAP value estimation and provide experiments demonstrating the superiority of these values in terms of differentiating among the different output classes and better aligning with human intuition than many other existing methods.

These techniques are generic and help interpret models deployed in multiple disciplines, such as healthcare, medicine, retail, and banking. A key factor concerning these techniques is that they have to be used on pre-trained models. We differ because we are trying to deploy techniques that are easy to interpret throughout the entire process, from data processing to problem classification.

2.3.3. Anomaly detection

Early works on fault detection suggested the use of time series *regression* methods and *Bayesian networks*. For instance, Barco *et al.* [52] produced an automated tool for troubleshooting mobile communication networks back in the days of 2G, relying on Bayesian networks to detect call drops. Khanafer *et al.* [53] followed up on proposing a method based on Bayesian networks to detect faults in UMTS systems, in which they apply different algorithms to discrete KPIs. Other works, such as [54], rely on a scoring-based system, in which the authors build the fault detection subsystem around labeled fault cases. These cases were previously identified by *experts*, using a scoring system to determine how well a specific case matches each diagnosis target. The work presented in [55] is based on a supervised genetic fuzzy algorithm that learns a fuzzy rule base and, as such, relies on the existence of labeled training sets. Indeed, most of the techniques proposed in the literature focus on using supervised machine learning algorithms [53–55]. In this paper we show that it is convenient to use unsupervised techniques to unveil hidden information in the input data, without restricting a priori the possible outcomes.

Other works make use of advanced mathematical and statistical tools. For instance, Ciocarlie *et al.* [56] address the problem of checking the effect of network changes via monitoring the state of the network, and determining if the changes resulted in degradation. Their fault detection mechanism uses *Markov logic networks* to create probabilistic rules that distinguish between different causes of problems. A framework for network monitoring and fault detection is introduced in [57], using *principal component analysis* (PCA) for dimension reduction, and kernel-based semi-supervised fuzzy clustering with an adaptive kernel parameter. To evaluate the algorithms, they use data generated by means of an LTE system-level simulator. The authors claim that this framework proactively detects network anomalies associated with various fault classes. These methods lack the flexibility of ML-based ones and, differently from our proposal, cannot be fully automated for a generic network context.

Recently, researchers are exploring the potentials of AI/ML for predicting and timely obviating network performance issues. For instance, Terra *et al.* [58] analyze how to apply existing XAI methods to identity the root causes of service level agreement violations in a sliced network, in a 5G context. Tang *et al.* [59] use instead ML to identify the root causes of exceptions in packet-over-optical networks, thus achieving what they define customized performance monitoring and troubleshooting. Furthermore, Wei *et al.* [60] point out that the class of AI that goes under the label of intent-based networking techniques can be used for troubleshooting of network services. For instance, available products like Cisco ACI and Spruce Network DeepFlow already offer network operators the capability to monitor and rise alarms and trigger troubleshooting actions in 5G and beyond 5G networks. However, they do not offer explainable nor interpretable methods and basically help in the self-optimization of a system using intent-based networking, which is limited to the scope of SDN/NFV network functionalities, while we will see in the reminder of this article that contextual information is key to identify anomalies rather than generic performance issues, and their causes. Moreover, existing AI/ML based approaches to troubleshooting are highly customized (e.g., for 5G functionalities or for optical networks) and require the collection of specific network features. Differently from our approach, these limitations make existing approaches unsuitable to be flexibly applied to new protocols and services.

Finally, it is worth mentioning that most of the existing proposals have been evaluated only through simulators, and require large datasets. They help to detect network issues, but do not contribute to interpreting the network behavior. By comparison, in our work, we use not-necessarily-abundant data collected in real operational networks and propose a fully automated ML-based methodology that leads to a straightforward interpretation of network behaviors. This includes identifying not only the occurrence of problems but also their root causes.

3 MONROE

The field of networking offers the possibility of gathering large volumes of information from network elements and end hosts. Analyzing these data is crucial to understand how networks perform under different usage patterns and adapt them to future requirements. This is particularly important for MBBs, which are the segment with the strongest growth forecast and higher variability in operating conditions. Two main challenges arise when trying to analyze the performance and reliability of MBBs: The difficulty of obtaining systematic data from reliable repetition of experiments on commercial operational MBB networks, and sifting through the big amount of variables that can be monitored and measured.

MONROE is a Europe-wide experiment oriented network counting on more than 200 custom measurement probes (or *nodes*), designed to enable collection and analysis of the characteristics of commercial mobile broadband networks and execution of discretionary experiments from external researchers.[1] The work carried out for this Thesis contributed to the design of MONROE services and to data handling, which will be presented in what follows jointly with the measurement architecture and some experiment examples, so as to show the potentials of MONROE.

The platform nodes operate under a wide variety of conditions, the nodes being deployed aboard trains, buses and delivery trucks, or inside residential homes and laboratories. Nodes are co-located in pairs, where one node connects to two mobile providers using customer-grade commercial subscriptions, and the other connects to a third operator and potentially to a WiFi network. Both nodes can connect to Ethernet where available.

The testbed performs periodic passive and active measurements and continuously monitors the status of the MBB networks through metadata collection. The collected metadata are centrally stored in a NoSQL database to ensure scalability past billions of records. We offer to the community the unique possibility of accessing our curated

[1] MONROE is a FIRE+ project funded by the European Union's H2020 research and innovation programme. For more information, please visit https://www.monroe-project.eu/

data-set through periodic data dumps, which enable data analysis across all the nodes and lifespan of the platform. Additionally, we encourage external experimenters to devise novel experiments and add to the diversity of MONROE open data.

The following is a list of the main characteristics and innovations of MONROE, which exposes software services and physical nodes to plan and perform MBB measurements, hence it is an *Experiment as a Service* (EaaS) platform.

Large-scale deployment in diversified scenarios: MONROE nodes are being deployed across Norway, Sweden, Italy and Spain, with external partners currently deploying additional nodes in Germany, Greece, France, Portugal, Slovenia and the UK. Some nodes have stationary locations in dense urban areas, while a significant number (more than 110 at the time of writing) operate aboard public inter-city trains, buses and delivery trucks. Whereas trains traverse large distances, sometimes at high speeds, buses cover urban areas. Both settings enable us to collect a unique data-set under mobility scenarios along the fix routes of those vehicles. Nodes aboard delivery trucks, which traverse both urban and rural areas without fixed routes, complement the previous data-set.

Open experimentation platform on commercial cellular operators: MONROE is an open platform that allows authenticated researchers to run their own custom experiments on commercial MBB networks. Researchers can then opt to add their data to the MONROE open data-set, increasing its diversity and allowing us to look past performance metrics and metadata. Notable examples are a Web performance experiment and video QoE measurements [61], which are being evaluated for inclusion in the set of periodic measurements run on the nodes. In addition to the actual data, experiment source code and supporting material for those wanting to create new experiments on MONROE are also openly available.[2]

Consistency and repeatability: MONROE provides a uniform hardware and software environment to measure and monitor MBB networks at fixed locations and times. Furthermore, the public transportation vehicles that host MONROE nodes ensure fairly repeatable routes for mobility experiments. Even more, they repeat the same itineraries several times a day at different hours (i.e., mixing peak and normal hours) and on different days (i.e., weekdays and weekends). This provides the data-set with a rich spatio-temporal dimension, which is key to enable the comparison of different measurements over different operators, places and times of day.

Metadata-rich data-set: Each MONROE node is instrumented to periodically measure the performance of its MBB providers. They continuously gather metadata, including, for example, location, signal strength and link technology for each network

[2] All stable pieces of open source code produced in MONROE are available on `github` at https://github.com/MONROE-PROJECT/Experiments, whereas a complete user manual is made openly available at https://github.com/MONROE-PROJECT/UserManual

provider. Additionally, several basic speed and network probing tests are executed periodically to asses network performance. Since MONROE does not involve real users (which usually entail privacy protection restrictions), rich metadata collection, including geo-temporal tagging, is possible, which enables the evaluation of mobile services under mobility. In particular, MONROE collection of data enables purely offline experiments for analysis of MBB network performance.

In the rest of this Chapter, we start by describing the design of the MONROE EaaS platform in Section 3.1. In Section 3.2, we first present the tools offered to experimenters and currently available *template* experiments, then we showcase the possibilities that MONROE opens by presenting a selection of experiments run by us and by several external groups. Those examples aim to entice other researchers to exploit the data gathered by our platform in innovative ways or to design their own experiments and so contribute to improve our overall knowledge on the behavior of MBB networks. We conclude the Chapter with a short discussion in Section 3.3.

3.1. EaaS platform design and implementation

The MONROE EaaS platform was designed with the purpose of collecting, storing and offering open access to large amounts of diverse mobile network data, and providing an EaaS platform for the execution of discretionary experiments by external researchers. Therefore, enriching measurement data with abundant context information (metadata), and enabling a wide variety of experiments, are the two key aspects that have steered the platform design since its inception. Figure 3.1 offers a high-level overview of the complete MONROE platform design.

Here we briefly present the platform components and focus on the processes of collection and storage of measurement results and the concrete implementation choices made during the platform design. The system design includes four main groups of components distributed across nodes and backend, as shown by the color code adopted in Figure 3.1.

The “red” component is responsible for MONROE default experiments, each using an isolated Linux Docker container [62]. Default MONROE experiments include, for example, periodic ping measurements for connectivity survey, HTTP downloads from a series of targets under our control, or Web performance measurements. The results of these default experiments and the collected metadata are transferred as JSON files to the main MONROE server via `rsync` over SSH channels. Once at the server, the JSON files are stored in a NoSQL database. Offline data analysis can happen both at the server side in the form of database queries or at the experimenter’s side (with custom applications) if further processing is required. Since data-sets are the main asset of the platform, we implement several backup and duplication mechanisms to provide data safety and access

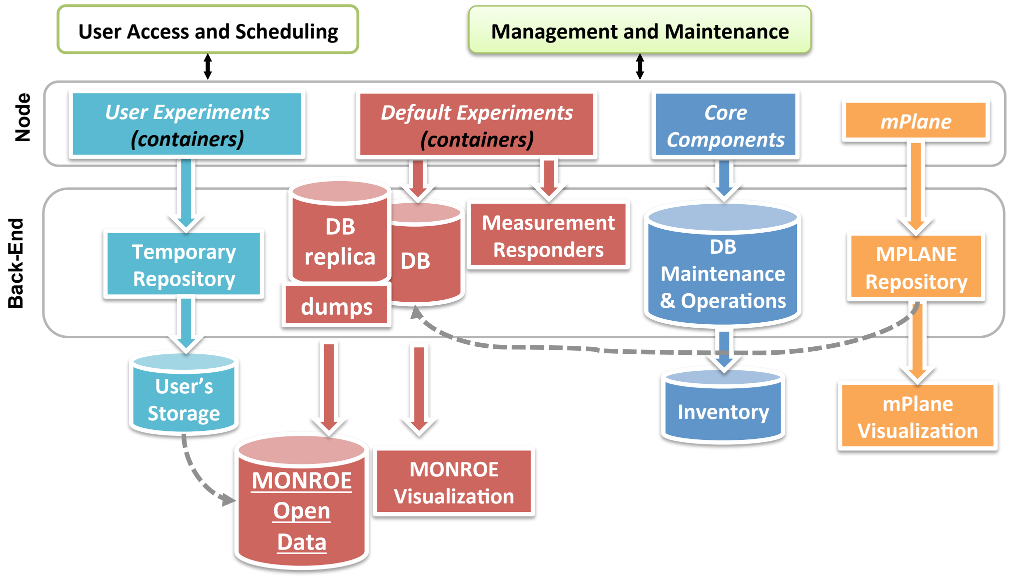


Figure 3.1: MONROE system design. Researchers access the system through the Web user interface and scheduler, or directly through the various repositories and data bases. Several passive (metadata, mPlane, etc.) and active (RTT, HTTP bandwidth, etc.) probes monitor network usage and performance continuously.

redundancy. A visualization solution facilitates the surveillance of the platform health and its available resources in near real-time.

Beside default experiments, MONROE allows authenticated external researchers to access the platform via the Web user interface and deploy their own custom experiments. This is the “azure” component of Figure 3.1. Separate storage for the results of user experiments is offered in a temporary repository accessible through the platform Web user interface. We encourage users to make their results public and include them in the MONROE open data-set.

In addition to default and external experiments, each node runs Tstat [63], a passive traffic analysis tool connected to the mPlane measurement platform [64]. Tstat generates a series of logs that the nodes send to the mPlane repository, from where users can consume the data using the mPlane visualization solution. This is the “orange” component in Figure 3.1. Note that Tstat data is also imported to the MONROE database, as shown in the figure.

A fourth component, the “blue” one in Figure 3.1, has been designed for dealing with node connectivity and software management of the platform.

As shown in the upper part of Figure 3.1, access to the platform is guaranteed to experimenters by means of a user access portal, and experiments are automatically loaded by a global scheduler that enforces and activates the Docker containers provided by the experimenters and carrying the experimental code. Thus, the entire architecture is transparent to the end-users, i.e., the experimenters. Moreover, platform maintainers have direct and exclusive access to the nodes and to the MONROE back-end.

3.1.1. Node instrumentation

MONROE nodes collect four types of information:

1. Metadata: This includes network parameters (RSSI, cell identifiers, link technology, etc.), node location and speed (GPS), node working parameters (CPU temperature, processing load, etc.) and node events (watchdogs).

2. Connectivity and latency measurements: Basic active measurements are run in a container that collects statistics on ICMP packets sent towards fixed destinations (UDP/TCP RTT will be added as future extensions).

3. MONROE and user experiments: Experimenters define Docker containers to run their measurements in isolation. Some containers are scheduled periodically to estimate available bandwidth, to track routes to and from specific targets in the network, etc. Other containers are scheduled upon the request of the experimenters.

4. Passive traffic monitoring: TCP flows are captured and analyzed by means of the Tstat measurement suit. MONROE nodes include a Tstat probe in a dedicated container, in which all MBB interfaces are monitored and where per-flow statistics are computed and subsequently published in the mPlane and MONROE databases.

The differentiation between the aforementioned types of data responds to their distinct natures and purposes. In that way, passive metadata can be gathered at the nodes with minimal impact on any experiments; thus, they are recorded on a continuous basis. Similarly, the passive mPlane Tstat probe, which produces low processing load, runs continuously. Background experiments such as end-to-end delay or round-trip time (RTT) measurements may create a moderate (controlled) interference with other experiments; however, the value obtained by gathering these data are worth their cost. Experimenters are made aware of those background experiments. Finally, some MONROE experiments such as bandwidth measurements might produce a higher impact on user experiments. Therefore, those experiments are not scheduled concurrently with any user's ones. Indeed, each user experiment runs in exclusivity with respect to experiments from any other users.

3.1.2. Data flows

Figure 3.2 shows the different flows of information through the platform, since it is generated in a node until it is collected and stored in our databases for later analysis. MONROE nodes implement a metadata distribution mechanism based on a publish/subscribe model. Experiments running in the nodes can subscribe to different information "topics" to monitor system status and events such as network interface (dis)connections, link technology changes or GPS location variations. This flexible design eases the implementation of each platform component as data producers do not need to

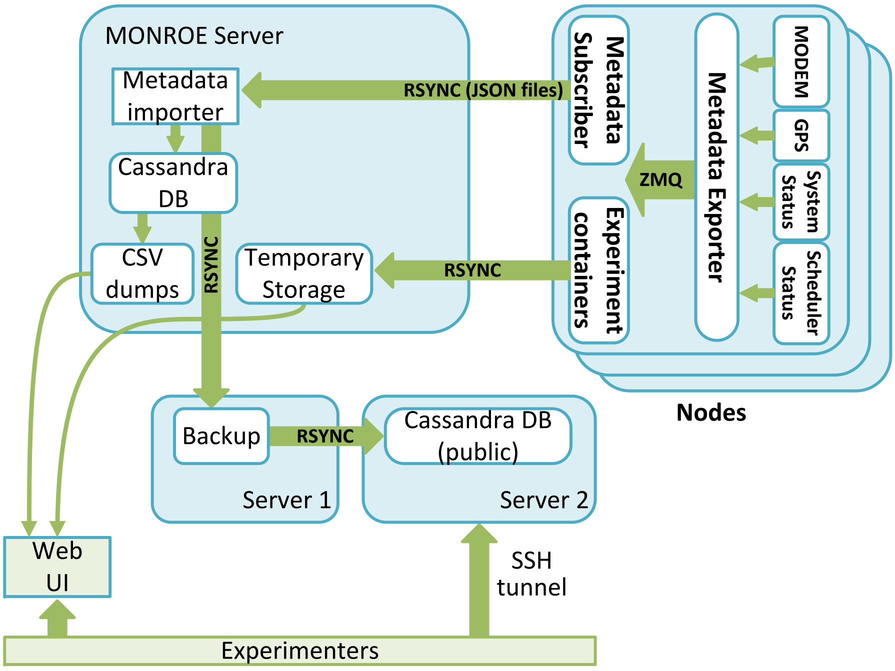


Figure 3.2: Flow of information in the MONROE platform.

keep track of their clients, and new data consumers can choose the information topics they are interested on without caring about the details of the producers.

Independently of their origin, all data items are transferred to the MONROE servers via `rsync` over SSH. Once at the server, each item is processed and stored according to its nature: Metadata, the results of the MONROE experiments and Tstat measurements, which arrived as JSON files, are stored in a NoSQL database, whereas the results of user experiments are temporarily kept at a repository for easy access through a Web user interface.

In the server, several scripts create backups of the database contents and a dump of the database in CSV format is produced daily; experimenters may use those 24-hour feeds if their experiments are focused on small periods of time. Furthermore, a secondary copy of the database is updated every day for direct access by external researchers. That secondary copy is not a normal database "replica" to avoid the risk that accidental (or malicious) modifications to the (open) database spread to the primary one. The daily CSV dumps are available for direct download to registered users through the Web user interface; access to the (secondary) database is provided to external researchers via SSH tunnels.

3.1.3. At the node side

At the node side, metadata distribution is implemented in a publish/subscribe pattern using ZeroMQ.[3] The metadata stream is available for experiments during their execution using the ZeroMQ subscription mechanisms. Metadata entries are generated in a single-line JSON format, which eases human analysis. Every data entry is labeled with a "topic" field; consumers may subscribe to the whole stream of metadata or just to some topics. The metadata subscriber module runs in the nodes and subscribes to all the topics, writing JSON entries to files in a special file system location. A synchronization process transfers those files to the MONROE server when no other active, periodic, or user-defined experiment is running.

Regarding node stability, several monitoring and recovery methods ensure that they remain online and capable of executing experiments. Node stability is ensured via lightweight virtualization (by means of Docker containers), thus guaranteeing a clean environment for each experiment. Several surveillance mechanisms (watchdogs) in the nodes can force a complete reinstallation of the operating system and environments if they detect system malfunctions such as filesystem corruption.

3.1.4. At the server side

Information received from the nodes in JSON format is stored at the server in a NoSQL database. The choice of a NoSQL solution was based on the need to permanently store a potentially very large data-set consisting of billions of entries. As a quick calculation to illustrate the scale of the data-set, RTT measurements are executed for each of the three MBB interfaces of each node every second. Therefore, $3 \times 3600 \times 24 \times 365 days \times 150 nodes = 14191e6$ entries could be stored in the database every year, only for RTT measurements, if they are run every second. Based on the concrete storage and access needs of MONROE, Apache Cassandra[4] was chosen as the system NoSQL database for its scaling abilities, both in performance and storage capacity. If the space available in a machine is exhausted, new space may be added simply by configuring a new replica. Additionally, Cassandra is a mature technology that offers access drivers for multiple programming languages and production-grade tools for data analytics, widening access options for researchers. Besides, several Python scripts produce a backup of the JSON files received at the server and a daily CSV dump of the database. Those results are transferred to a backup server that provides off-site backups. The copy of the database accessible to external researchers is hosted in an independent server, thus avoiding performance interferences with the main database.

[3] ZeroMQ distributed messaging: http://zeromq.org

[4] The Apache Cassandra database: http://cassandra.apache.org

3.1.5. Access to data

The metadata produced by the nodes can be accessed in several ways. First, experiments may access the metadata stream during execution using the ZeroMQ subscription mechanisms. In this way, they can monitor and react to events such as interface reconnections or link technology and signal strength changes for each MBB at run-time. Second, researchers may access the database (or the CSV dumps) to correlate their results with the metadata matching by the corresponding timestamps. As an example, the results of an experiment may be related to the network conditions during its execution, even if at that time not all the metadata was checked online. Researchers may also import the CSV dumps into their own tools for more specific data analyses.

3.1.6. User access and experiment scheduling

MONROE enables user access to the experimental platform through a user-friendly interface built on an AngularJS-based Web portal. The platform is open, although authentication is required. In particular, as part of the MONROE federation with the Fed4FIRE initiative of the European Commission,[5] MONROE user access follows Fed4FIRE specifications in terms of authentication and provisioning of resources. Hence, the MONROE portal allows to access the MONROE scheduler, which is a server in charge of setting up the experiments without requiring the users to directly interact with the nodes (i.e., no SSH access to the node environment). The scheduler ensures that there are no conflicts between users when running their experiments and assigns resources to each user.

The scheduling system consists of two parts. A scheduling server runs on a MONROE server behind an Nginx proxy and uses an SQLite 3 database to store user roles, node and experiment status, and schedules. In addition to Fed4FIRE-compatible APIs, it offers a REST API that can be accessed through the Web user interface or directly through the Nginx proxy if users develop their own access scripts. The client part of the scheduler runs on MONROE nodes. It periodically contacts the scheduler in the server to send "heartbeats" and traffic statistics, and check for new schedules for the node. When new schedules are available, the scheduler preloads up to three containers, depending on criteria such as available storage in the node and time until schedule. It also schedules the start and stop times of each container using operating system functions. When the time to execute a new container arrives, the operating system executes the container using the Docker tools. Finally, the scheduler monitors the experiments to check if they exceed the allocated resources and to transfer any result files and inform of result codes.

[5] Fed4FIRE, testbed federation for Future Internet Research and Experimentation (FIRE): http://www.fed4fire.eu

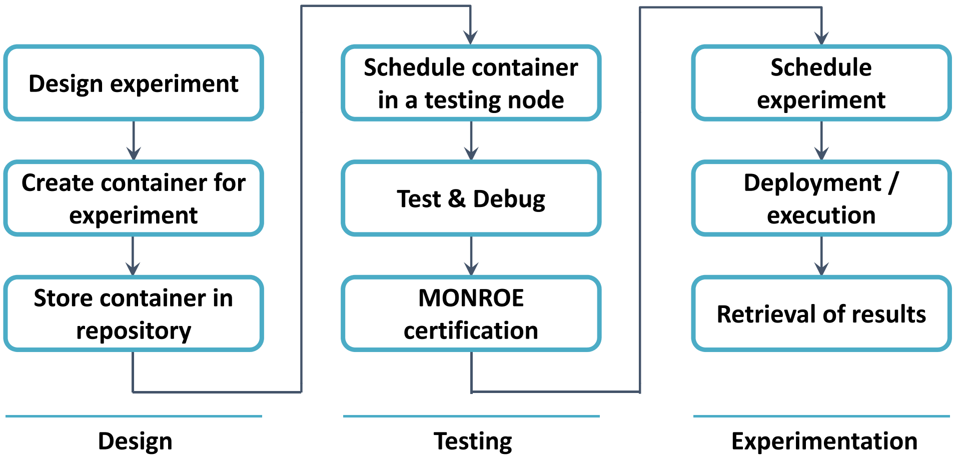


Figure 3.3: Experiment workflow covering the design, test and experimentation phases.

3.1.7. Experimentation workflow

Figure 3.3 shows the general workflow of the experiments executed on MONROE nodes. The first step is to design the experiment selecting the appropriate tools. The required files have to be collected in a Docker container, which is submitted to a repository. MONROE offers a set of dedicated testing nodes that can execute containers from any public repository. Once the experiment is ready, it undergoes a certification process in which MONROE administrators check that it is generally safe for execution and move the container to a private repository. Deployed nodes (i.e., real experimentation nodes) can download containers only from the MONROE private repository. Container execution can be scheduled as many times and on as many nodes as required, always subject to quota availability. Using the platform Web interface, users can monitor the progress of all their experiments, including repetitions on multiple nodes. Finally, the results can be downloaded directly from the platform Web page.

3.2. Experiments

In this section, we list and describe the EaaS templates currently available in form of Docker containers and ready for experimentation with the MONROE platform, as well as a number of their use cases for MBB evaluation.

3.2.1. Experiments currently available as services

There are many experiments available as services on the MONROE platform within an EaaS framework. Figure 3.4 lists these experiments with respect to their origin and characteristics.

Consortium experiments are provided by the MONROE Consortium and are all

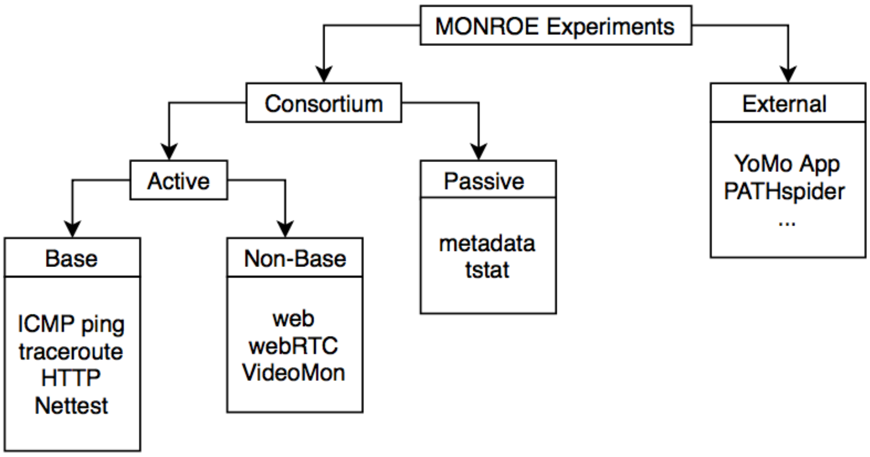


Figure 3.4: Experiments currently available as services that can be run on the MONROE platform, within an EaaS framework.

available on `github`, as mentioned in the introduction, jointly with detailed instructions on how to configure and run the experiments. For each experiment and the Docker container implementing it, a template is prepared and provided for running the experiment from the MONROE Web interface with a single click, without the need to set any configuration parameters. The default parameters can be modified at will, as documented in the MONROE user manual.

The experiments can be passive or active. Passive experiments collect information in the background without generating additional traffic, whereas active experiments perform new measurements by generating data traffic. Passive experiments run continuously (periodically). Active experiments can either be run periodically with a lower frequency than the passive experiments (those are referred to as “base” experiments), or be available for running at will without a regular schedule (“non-base experiments”).

In addition to the consortium experiments, the platform was also used for research and experimentation by 27 external groups from academy and industry, who run both passive and active experiments.

In what follows, we describe the design and implementation of the most prominent experiments produced by the MONROE consortium and provided as EaaS, and give two examples of experiments designed by external experimenters.

3.2.1.1. Metadata collection

MONROE nodes passively and continuously generates metadata. Table 3.1 illustrates the metadata “topics”, which are streamed to subscriber entities within the node using ZeroMQ, as previously explained in Sections 3.1.2 and 3.1.3. Metadata are collected and stored in a database for post processing. However, experiments running containers can also have their containers subscribed to any of the metadata topics and use them during

Table 3.1: MONROE metadata topics

Class	Type	*Examples*
Node	Sensor	*CPU temperature*
Node	Probe	*Load, memory usage*
Node	Event	*Power up, reboot*
Device	GPS	*GPS coordinates*
Device	Modem	*RSSI, link technology, cell ID, IP address*
Experiment	RTT	*Ping RTT*
Experiment	Bandwidth	*HTTP download throughput*

their experiments or store them jointly with their results, to easy the joint post processing of data and metadata pertinent to a given experiment. Upon a variation in a monitored value, a new message is sent to subscribers only, so the metadata generation uses limited resources.

3.2.1.2. TCP flow analysis

One of the Docker containers always present in a MONROE node runs TCP flow analysis in near-real time using Tstat. Tstat is a powerful passive monitoring tool that rebuilds TCP flows reporting more than 100 flow descriptors (e.g., client and server IP and port, RTT, number of retransmissions) and more than a thousand packet level metrics [63]. Therefore the container implements a passive traffic probe that provides insights on the traffic patterns at both the network and the transport levels, offering additional information on the traffic each interface exchanged during an experiment. This container runs continuously and does not interfere with other experiments. Moreover, experimenters can use Graphite to easily navigate through offline logs and store a dashboard showing relevant data within an adjustable time window.[6]

3.2.1.3. End-to-end delay statistics

This is a base experiment running active and lightweight measurements. It consists in a simple container that pings continuously a few remote targets and records ICMP ping statistics. A variant of this container is also available, in which UDP is used instead of ICMP. Despite being very simple, this experiment gives fundamental information about the status of the network and its congestion level.

3.2.1.4. Route monitoring

MONROE incorporates active traceroute measurements in the set of base experiments, to study routing and to identify middleboxes. The MONROE traceroute experiment

[6] Graphite documentation: http://graphite.readthedocs.org/en/latest/index.html

aims to compare routing from nodes in different countries and, inside a country, different operators (some of our measurements are performed with SIM cards in roaming that show home-routing patterns). By using Paris Traceroute rather than a simple and legacy traceroute application,[7] these experiments also allow identifying middleboxes and their differences between operators and countries.

3.2.1.5. Webpage download

To assess basic Web performance figures, one of the experiments available as service in MONROE, WebWorks, is an active on-demand experiment using the Firefox browser in headless mode allowing to run in a node with no need of a monitor. WebWorks is built on top of the Selenium Web automation framework.[8] Selenium provides a Web driver that can interact with Firefox as a regular user would. During a page visit, WebWorks uses the HAR export trigger add-on to log Firefox interactions with the page as JSON-formatted archive file called HAR (HTTP Archive).[9] WebWorks uses the HAR file to derive a number of Web performance metrics such as DNS resolution time, TCP connect time, object receive time specific to various objects in a page. Besides WebWorks tracks three other metrics, namely Page Load Time (PLT), Byte Index (BI), and Object Index (OI). PLT is primarily based on OnLoad event triggered by the browser when all objects on a page are loaded. OI and BI are time-integral QoE metrics derived from the HAR files [65]. They are computed from the arrival time of all objects in the webpage waterfall. OI tracks the time at which the content of the page is retrieved, taking into account all external images, stylesheets and scripts needed to render the page. BI operates in the same way, but weights objects by their size. For both, a higher value indicates higher page load time and higher delays at the reader's browser. WebWorks measures Web performance against multiple popular targets, enabling, for example, the tracking of PLT and other metrics and their correlation with metadata information.

3.2.1.6. HTTP download

This is another active base experiment that is periodically scheduled in all nodes. The container tests HTTP download rates using the various available versions of HTTP, and generate statistics about large file downloads. Being data-consuming, this test is not aggressively scheduled, although it is needed to complement the statistics on delay/RTT studied by means of tiny ping packets and on short-lived flows collected with the webpage download experiments described above.

[7] Paris Traceroute network diagnosis/measurement tool: https://paris-traceroute.net

[8] Selenium browser automation: http://www.seleniumhq.org

[9] HAR export trigger: http://www.softwareishard.com/blog/har-export-trigger

3.2.1.7. Network speed tests

MONROE-Nettest is a configurable tool for data rate and latency measurements, intended for the study of speed in MBB networks, using active experiments. We choose RTR Multithreaded Broadband Test (RMBT) by Netztest[10] as the codebase for our client implementation since this is a tool used by most network regulatory authorities in Europe for their crowdsourced measurement applications. Adopting a user experience oriented approach for measuring data rate, these solutions use TCP-based testing with multiple parallel flows. Configurable parameters of the client include the number of flows for downlink and uplink, measurement durations, and measurement server. For the server side, we make sure to keep compatibility with the RMBT, and use the server code from the open-source Open-RMBT project,[11] with only minor changes. We have deployed a network of MONROE-Nettest servers in Europe, including Germany, Norway, Spain, and Sweden for large scale experimentation. MONROE-Nettest[12] is run as a base experiment in the MONROE platform, so it is run periodically on every node and every connected MBB network.

3.2.1.8. Adaptive streaming over HTTP

MONROE uses a variant of AStream, which is an open source software written in Python to implement 3 different rate adaptation algorithms for evaluating MPEG Dynamic Adaptive Streaming over HTTP (DASH).[13] We have adapted the existing AStream framework to the MONROE platform with slight modifications, providing a suitable Docker container which integrates a wrapper. Therefore, this is an active type of experiments, which currently run as a non-base MONROE container. However, this experiment will soon be run as a base experiment within the VideoMon container,[14] which is a combination of the consortium experiment AStream with the external user experiment YoMoApp (more info in Section 3.2.1.9).

3.2.1.9. Video QoE with YoMoApp

YoMoApp is an application for YouTube performance monitoring, which allows analyzing mobile network performance with respect to YouTube traffic [61]. It also serves developing optimization solutions and QoE models for mobile HTTP adaptive streaming. The application has been developed by external MONROE experimenters to extend MONROE into the domain of QoE with the design and implementation of a measurement tool for YouTube video streaming sessions. YoMoApp gathers statistics on

[10] RMBT specification: https://www.netztest.at/doc/

[11] https://github.com/alladin-IT/open-rmbtcommitdfc008de71e321c863716b0d34208159b140c653

[12] https://github.com/MONROE-PROJECT/Experiments/tree/master/experiments/nettest

[13] https://github.com/pari685/AStream/

[14] https://hub.Docker.com/u/videomon/

initial delay, video adaptation over HTTP, HTTP request and response information, and stalling occurrences [66].

3.2.1.10. Path transparency

This is another example of MONROE container and experiment developed and provided by external experimenters. The container uses [67] to detect the presence of middleboxes over point-to-point paths. In addition, it tests the feasibility of deploying new protocols in the Internet while quantifying the impact of path impairments.

3.2.2. Selected measurement studies

Next, we present some of the most interesting studies that have been conducted on MONROE using the previously described experiments and/or the MONROE data-set, which, at the time of writing, contained more than 2102 M metadata entries, 4230 M RTT and 107 K bandwidth measurements, 102 M Tstat entries and more than 50 K experimenter results.

Studies on the MONROE platform can be passive or active. *Passive studies* analyze and use the curated MONROE data-set, which contains metadata, the results of the default experiments and the results of experiments shared by their owners with the broader community. They can perform queries directly on our NoSQL database or process the CSV files that are generated daily (e.g., for more complex analyses on smaller amounts of data). Those experiments can use the whole range of MONROE data, since the moment it started to collect information, and for all the nodes in all the countries, and can be repeated at any point in time. *Active studies* are executed on MONROE nodes via explicit scheduling. They use the experiment services provided as Docker containers and schedule them on real nodes through the platform Web user interface. Those experiments can consist of any software compatible with the container architecture and use all networking resources available in the nodes at the moment of execution, subject to user quotas availability.[15] Experiments can be repeated as desired to verify the consistency of the results or to analyze changes on network behavior along time. The new data generated by active experiments may become part of the data-set available for passive experiments.

3.2.2.1. Studies by the Consortium

In what follows we describe some of the key studies conducted by using the available MONROE experiment containers, and show samples of our measurement campaigns. However, here we only focus on showcasing the kind of experiments that can be performed and put no emphasis on performance figures and comparisons between services offered by different operators. Therefore, we do not provide a complete and exhaustive set of

[15] For fairness, MONROE users receive a share of the platform resources.

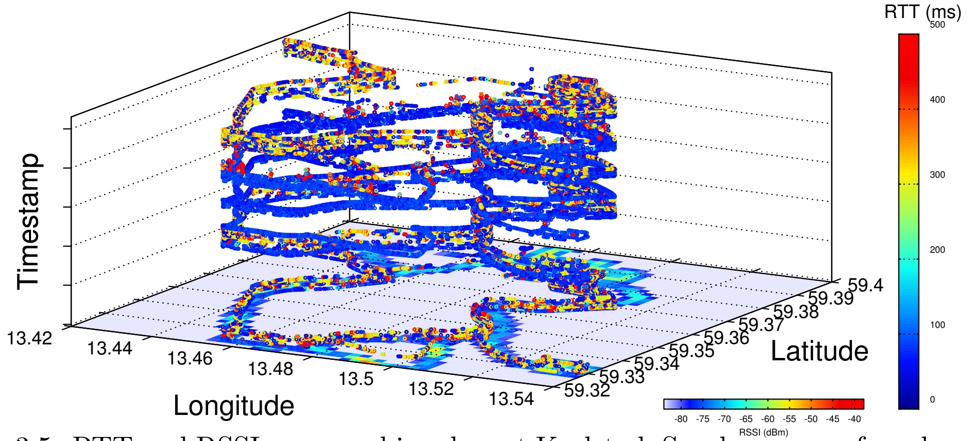


Figure 3.5: RTT and RSSI measured in a bus at Karlstad, Sweden, over a few observation days. Average RSSI values are shown on the XY plane. Individual RTT measures are plotted on the Z-axis using their relative timestamps as height to visualize successive laps.

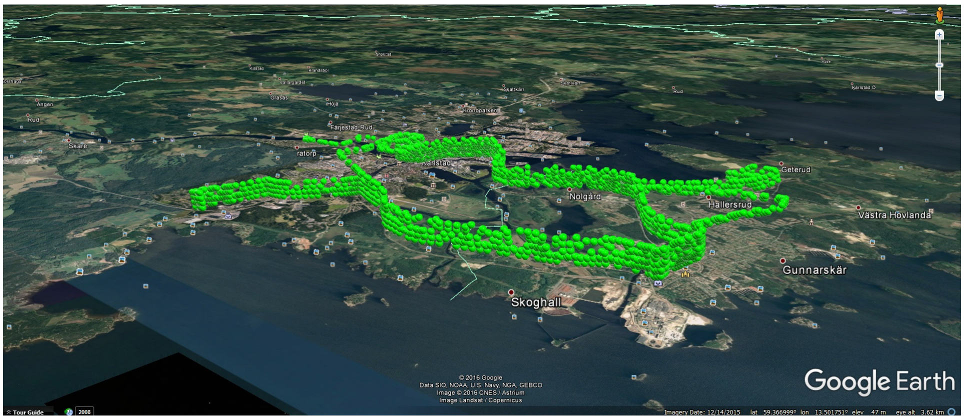
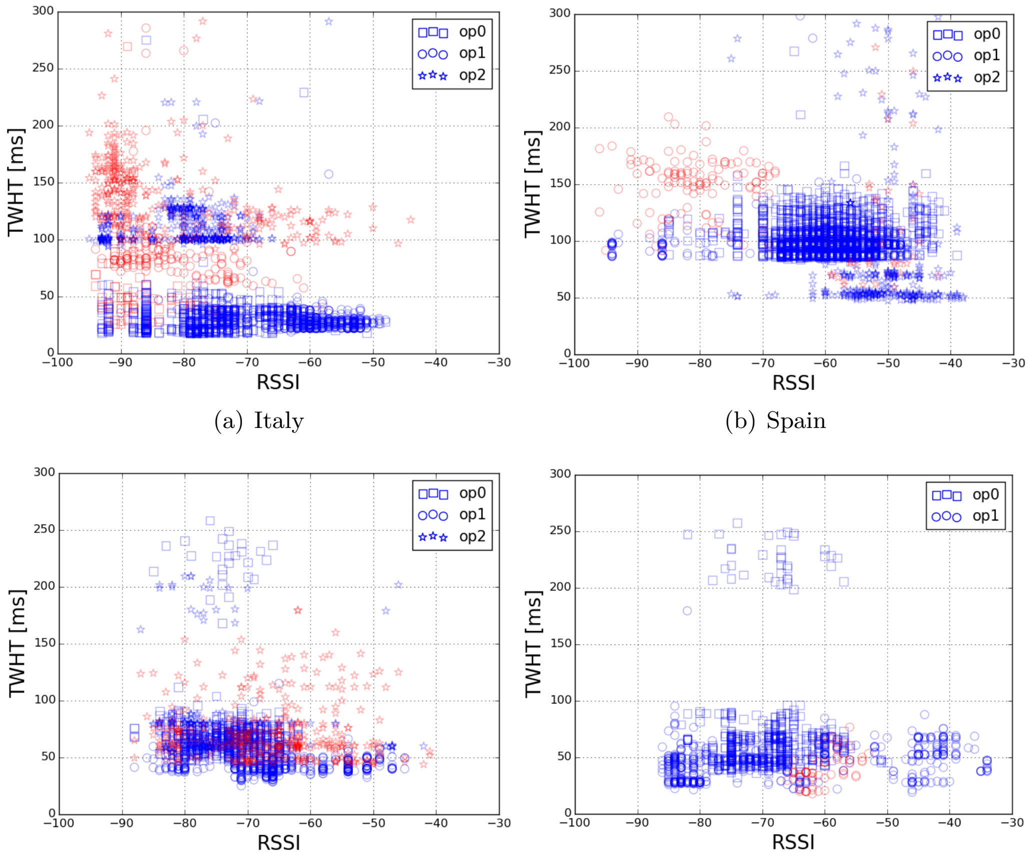

Karlstad
Nolgård
Geterud
Hallersrud
Västra Hovlanda
Gunnarskär
Skoghall
Google Earth
© 2016 Google
Data SIO, NOAA, U.S. Navy, NGA, GEBCO
Image © 2016 CNES / Astrium
Image Landsat / Copernicus

Figure 3.6: This representation of link technology for the bus at Karlstad reveals that 4G coverage is consistently available for the complete route during the analyzed period.

experiments for all operators and all countries in which we have run the measurements, and we anonymize our measurements with respect to operator names. The results shown in what follows are not representative of the full coverage and service offered by operators across Europe, although the platform could be used to pursue such goal.

Metadata/QoS analysis to build coverage and latency maps. MONROE deployment in public transportation vehicles enables evaluation of MBBs on wide urban mobility environments. Route predictability provides high confidence, whereas measurements taken at similar positions on different hours allow comparing the behavior of the MBBs at different times (e.g., rush hour versus normal hours).

Figure 3.5 follows the typical route of a bus around Karlstad (Sweden), showing the measured RSSI (signal strength) and RTT (ICMP ping). The different laps along several days are represented vertically ascending to ease the visualization of the dense information obtained. Figure 3.6 shows the negotiated link technology for the same route. The analysis of the collected data (signal strength, link technology and measured delay) gives insights



(a) Italy

(b) Spain

(c) Sweden

(d) Norway

Figure 3.7: TCP three-way handshake times (TWHT) obtained using the HTTP download experiment for bandwidth measurement with different operators versus the RSSI reported in MONROE metadata. Blue and red correspond to 4G and 3G samples, respectively.

into the performance perceived by users during their bus trips. Such information might then be used by network operators to improve the service offered to commuters.

Based on the same data-set and on theory and observations that show that fading follows a Rice distribution under line-of-sight conditions, while it follows a Rayleigh distribution otherwise [68], we are currently developing a method to infer which distribution yields a better fit for experimental data, potentially providing information to operators to optimize the location of base stations.

Traffic analysis and network monitoring with Tstat. We have used Tstat to study the performance of TCP flows as observed by the MONROE nodes. As an example, Figure 3.7 shows a correlation between three-way handshake time as measured by Tstat, and RSSI from the metadata, illustrating the many possibilities that MONROE creates for cross-domain data analysis.

Operator benchmarking with cross-country performance. MONROE enables

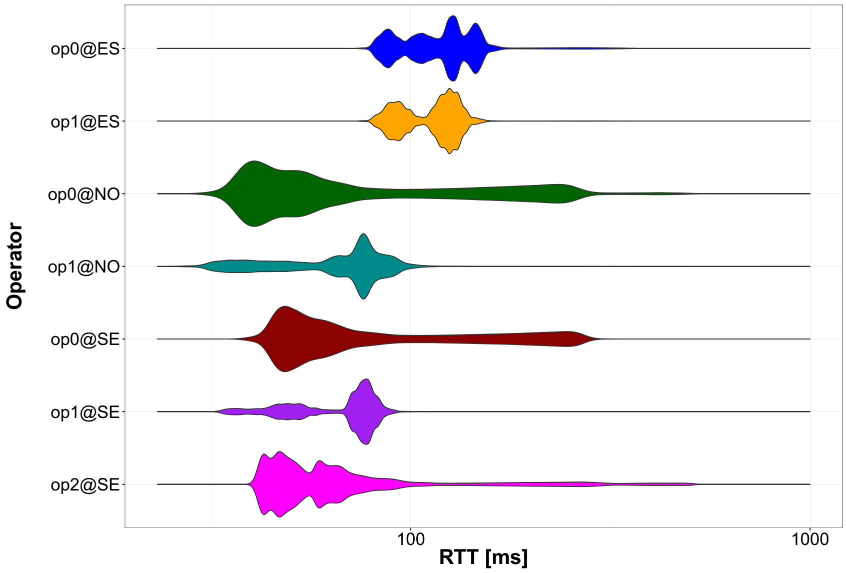
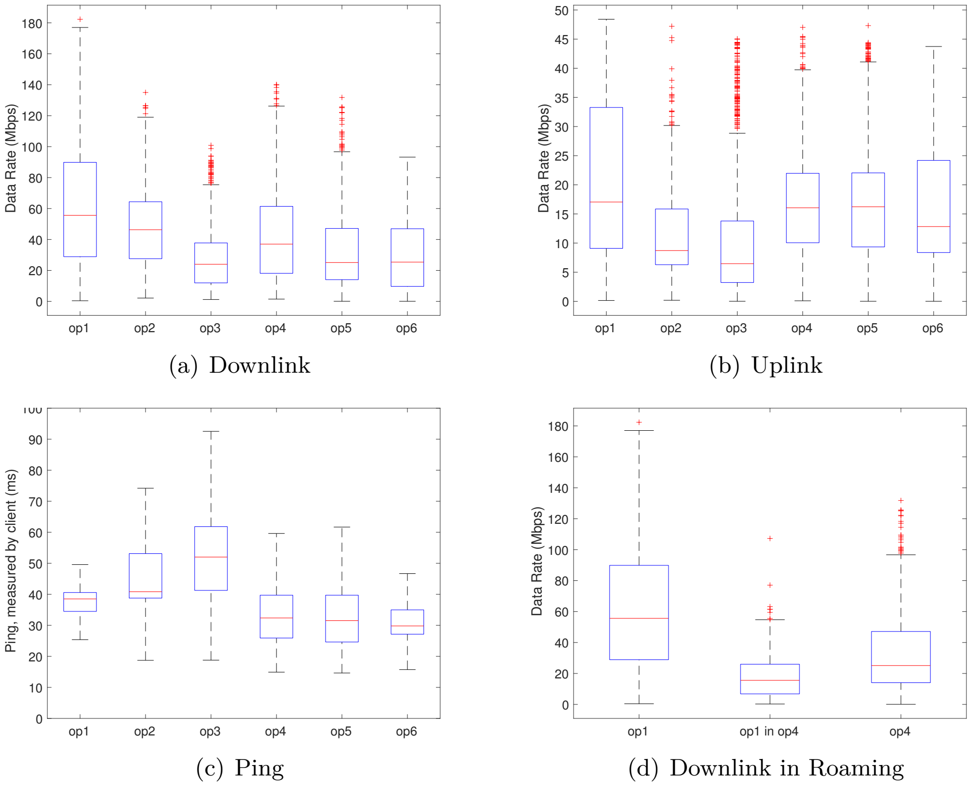


Figure 3.8: Violin plots of the RTT measurements for different operators in Spain (ES), Norway (NO) and Sweden (SE).

comparison of different operators (in terms of network characteristics and user-perceived application performance) in and among countries. For this purpose, multiple MONROE services, such as the ICMP ping container, and the Nettest container can be used.

Figure 3.8 shows a violin plot for the RTT samples collected (using ICMP ping) during one week with 30 stationary nodes for 7 different operators in 3 countries. Each "violin" shows the probability density of the RTT at different values; the higher the area, the higher the probability of observing a measurement in that range. Nodes in Norway and Sweden exhibit lower delays than nodes in Spain because they are closer to the target measurement server, which is hosted in the MONROE backend in Sweden. Interestingly, measurement variance is much higher than in fixed networks, showing that MBBs introduce complexity even for such basic tests as RTT monitoring. For example, RTT measurements exhibit typically a multimodal distribution that corresponds to the different access delays faced by different radio access technologies (e.g., 3G vs. 4G). MONROE repetitive measurements enable correlation with time, location and context conditions such as variations in signal strength.

It is also possible to benchmark operators using the MONROE-Nettest container. Running as a base experiment, this container has provided more than 850000 measurements over stationary and mobile nodes in Norway and Sweden since June 2017. Figure 3.9 presents an overview of the downlink and uplink data rate, as well as latency values for stationary nodes and 6 operators (3 in Sweden, 3 in Norway), including an example case of roaming. For each operator camping on its own network, we use the MONROE-Nettest server in the corresponding country (Figures 3.9a–c). The roaming example in Figure 3.9(d) shows the downlink data rate for operator *op1* (Sweden) camping



(a) Downlink

(b) Uplink

(c) Ping

(d) Downlink in Roaming

Figure 3.9: MONROE-Nettest base experiment results.

on *op4* (Norway), compared with the native downlink data rates for *op1* and *op4* from Figure 3.9(a). For this comparison, we had client nodes in Norway using *op1* SIMs, and the measurements have been conducted against the MONROE-Nettest server in Norway.

Investigating the speed of mobile broadband. In [69] we present our experience estimating the download speed offered by actual 3G/4G networks. For that experiment, we analyzed data from 50 nodes in 4 countries over 11 operators during more than two months, using the Tstat container. The conclusion of that study is that measuring the performance of MBB networks is quite complex as different network configurations such as the presence of NATs or Performance Enhancing Proxies (PEPs), which do vary over time, have a significant impact on measurements.

We have made similar observations using the active MONROE-Nettest container, where the effect of measurement methodology has proven to be a key factor affecting reported data rates. Currently, we have identified 3 main aspects of active measurements that influence data rate as: number of parallel TCP flows, measurement duration, and server location.

Web performance. Web performance is assessed by means of the WebWorks experiment described in Section 3.2.1.5. In [70] we have shown preliminary results from our experiments on Web page load time (i.e., PLT) and proxy identification over mobile broadband networks. There, we use a headless browser to fetch two popular websites from 37 nodes operating in four countries and using 11 operators. As an example, we observe

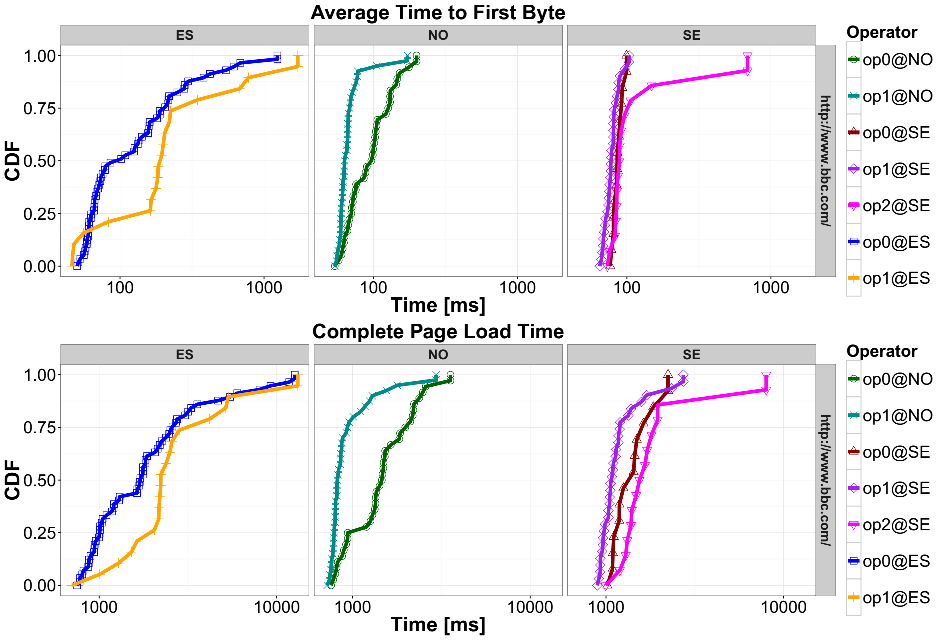


Figure 3.10: Average Time to First Byte and Complete Page Load Time for some operators in Spain (ES), Norway (NO) and Sweden (SE) for www.bbc.com.

large variations of PLT for the same website between Sweden and Norway. In that work we also report results on identification of PEPs in MBBs.

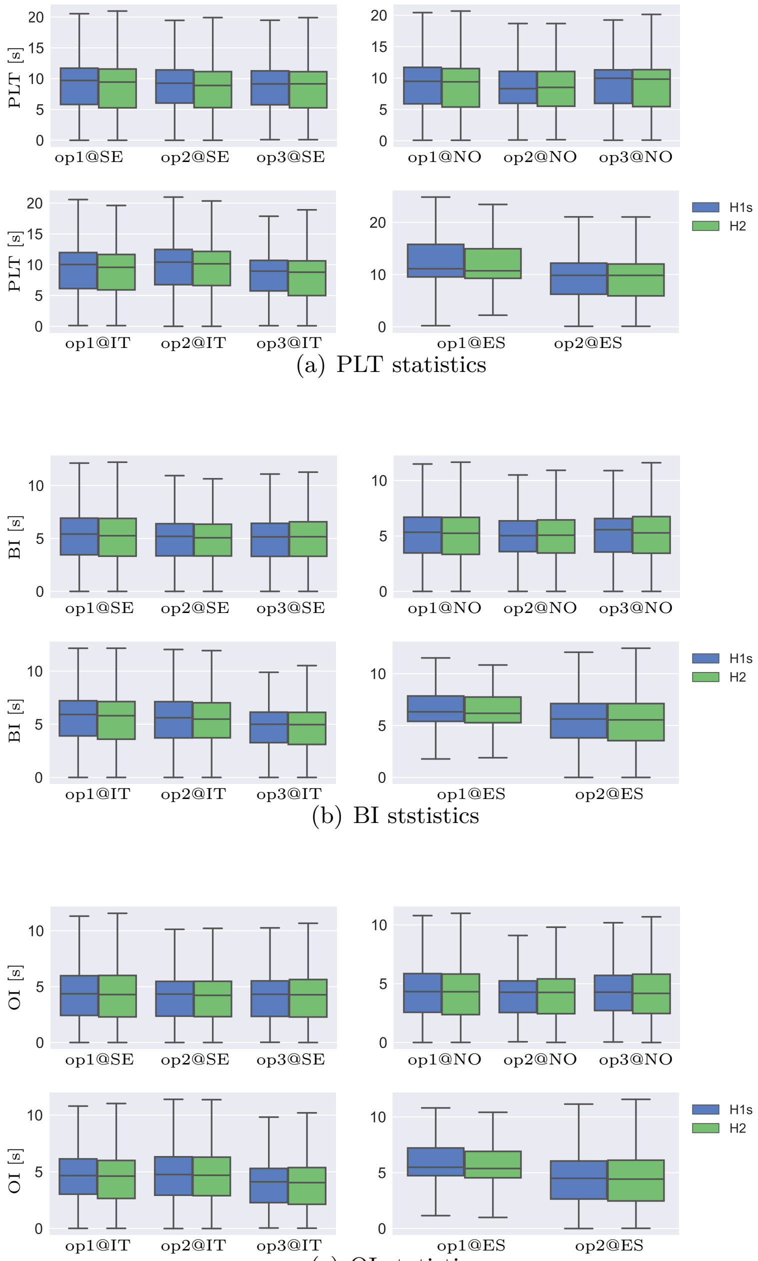
In Figure 3.10, we present the CDFs of the complete page load time and average time-to-first-byte for www.bbc.com broken down per country. Interestingly, for the Spanish operators we detected multiple DNS iterations, which partially account for their higher time-to-first-byte values.

If we consider multiple websites, we obtain the results shown in Figure 3.11. In there, we show not only the PLT metric, but also the two time-integral metrics computed by WebWorks, namely OI and BI. Such metrics show that overall Web performance is similar across different countries and operators, with only slight variations. At this aggregate level, we also observe similar performance between HTTP versions (indicated in the figure as H1s in case of version 1.1 with TLS, and H2 in case of version 2.0).

3.2.2.2. Studies by external experimenters

Here we give some specific examples of experiments designed by external users and deployed on the MONROE platform. Note that, thanks to the openness of our platform, some of the described experiments have been built on top of MONROE, by extending our nodes with additional hardware and/or software. For detais on extensions and results obtained by experimental researchers, in what follows we give specific pointers on a case-by-case basis.

Software radio extensions. The SOPHIA project has developed an extension to enhance MONROE nodes with software radio capabilities. In [71], its members present detailed performance measurements of LTE networks to illustrate the potential benefits



(a) PLT statistics

(b) BI ststistics

(c) OI statistics

Figure 3.11: Country-wise per-operator overall webpage download performance.

and new possible passive measurements obtained by decoding the control channels of LTE.

Forecasting LTE cell congestion. In [72], the authors try to forecast the average downlink throughput for LTE cells using data collected from multiple MONROE probes and to apply that knowledge to self-organizing network strategies to shift coverage and capacity according to predicted demand. This group updated some MONROE nodes to address the benchmarking of voice calls, showing the flexibility of the platform nodes.

Available Bandwidth measurement on SDN deployments. In [73], the authors employ MONROE as a testbed to study the complexity of available bandwidth estimation using SDN-based active measurements. They conduct their experiments using one node in each of the four main countries of the project. Their ongoing work tries to improve the accuracy and reliability of existing tools, using the MONROE testbed to isolate and better understand different aspects of the measurement process.

Designing application performance with MBB analytics. The authors of [74] use the radio parameters measured by MONROE nodes to determine the best application protocol for a service, identifying the most suitable key performance indicators to characterize the network state. These types of works are very relevant to close the gap between network performance measurements and user experience. Interestingly, the authors see an opportunity on the data generated by other experiments running in the platform (and made openly available by the respective researchers) as a means to obtain additional data points for their own investigation.

Surveying DSCP modifications in mobile networks. MONROE is used by a group of researchers in [75] to conduct a survey on path-level treatment of DiffServ packets in MBB networks and identify behaviors that potentially violate the IETF specifications. DiffServ enables the classification of traffic into QoS classes via usage of the Differentiated Services Code Point (DSCP) field in the IP packet header. Using MONROE to analyze the behavior at the edge mobile network, they find that there is a high probability that the corresponding fields are overwritten in the first two network hops.

Path protocol transparency. [67] is a tool developed for A/B testing of path transparency. It allows testing the feasibility of deploying new protocols in the Internet and quantifying the impact of path impairments and of middleboxes. In [76], the authors, in collaboration with part of the MONROE consortium, present the results of adapting PATHspider, to the realm of commercial mobile networks using MONROE nodes deployed by themselves in the UK. Among their conclusions, the most relevant is that MBB networks provide a considerably different environment—and therefore very valuable—with respect to the one provided by the cloud access points that PATHspider was using in the past.

3.3. Discussions

In this Chapter, we have described the unique EaaS offered by MONROE and discussed how it allows to collect, curate and make available valuable and uniquely rich and open data-sets to the community. We have focused on how MONROE helps to improve the knowledge on the usage and behavior of current and future commercial mobile broadband networks. We have also explained the main design characteristics of the platform that make it unique and how, from the generation of data at the nodes to their storage in a NoSQL database that can scale past billions of records, MONROE offers the unprecedented possibility of data analysis across all the nodes and lifespan of the platform. We have presented several and key experiments designed by the MONROE Consortium and by external experimenters. Eventually, to illustrate the potential and flexibility of the platform, we have presented samples of results from our own experiments in which the author of this Thesis was mainly involved in terms of developing the platform and the tools used, and providing support to the experimenters, while experiments specifically designed and run for this Thesis will be described later. As shown, the MONROE platform has been also used by several research groups that have been granted access to the platform.

Part II

Network Protocols Performance Evaluation

Over the past decade, TCP and UDP prevailed as the dominant networking protocols, but these two protocols could not cope with ever demanding mobile market as the years passed. Separately, UDP and TCP cannot handle Real-Time Communication (RTC). However, in tandem, they gave birth to Real-Time Communication (WebRTC). WebRTC proposes integrating video services in Web browsers based on local tools. In turn, such tools are based on well-known Web technologies, able to incorporate audio, video, and data transfer operations of RTC into a standard webpage. The evolution does not stop with WebRTC, but with another additional protocol: Quick UDP internet connection (QUIC), designed mainly to solve problems encountered with TCP and dominate internet traffic for years to come. Both developed and maintained by Google since 2011 and now officially standardized by the IETF. Since these protocols are fairly newish, a need to performance asset them is essential in their early development cycle, not only in simulated/controlled environments but also in real operational cellular networks.

In Part II of the Thesis, we investigate performance figures of WebRTC and QUIC in real operational mobile networks in static and mobility scenarios. Chapter 4 mainly focuses on WebRTC performance evaluation in cellular networks across different European countries leveraging the MONROE testbed. We developed and made available for free a Docker container for running experiments alongside the data-set collected based on the WebRTC open-source code to gather stats, for which so far little experimental work existed at the time of writing.

Chapter 5 centers on the experimental assessment of QUIC and congestion control schemes in cellular networks. In there, we derive a method for evaluating the performance of QUIC by leveraging two open-source implementations. All the stats collected are available as open-source for the interested experimenter. We assess both implementations with different congestion control algorithms such as Copa, Bbr, newReno, Cubic in different real-world networking scenarios.

4 Performance Evaluation of WebRTC

WebRTC proposes to easily integrate video services in Web browsers, based on local tools [10]. In turn, such tools are based on well-known Web technologies, able to simply integrate audio, video, and data transfer operations of the real-time communication protocol (RTC) into a normal webpage. The WebRTC project[1] was first introduced by Google as an open source project, and then other software developers and telecom vendors joined, which has led to integration of WebRTC into commercial browsers like Chrome, Opera and Firefox [11, 12].

Since offering video services and multimedia channels is a *killer application* for MBB networks, WebRTC and similar projects impose stringent quality requirements on such networks, that are nowadays evolving from 4G to 5G under the pressure of a steadily increasing number of mobile users [24]. Therefore, there is now a strong need for objective information about MBB performance and reliability to support video and multimedia mobile services. Thus, various initiatives have arisen, among which the US FCC's Measuring Broadband America initiative [24] and MONROE, to monitor and assess MBB performance.

We focus on WebRTC performance figures in mobile environments, for which so far little experimental work exists. In fact, other works on assessing WebRTC performance figures are currently sprouting, but they have so far only investigated basic properties in controlled environments. For instance, the authors of [14] used a cloud-engineered automatic testing tool for WebRTC, although they have not tested the service offered by mobile operators and core networks. Similarly, the authors of [15] have experimentally tested one-to-many communications over WebRTC (namely "simulcast"), although their experiments are limited to a gigabit LAN environment.

In contrast, for our work, we leverage a large-scale on-line measurement platform and focus on users connecting through MBB networks only. Specifically, we use the above mentioned MONROE platform, which has been designed and is currently operated in the frame of a European project aimed at providing multi-homed, independent,

[1] http://www.webrtc.org/

large-scale monitoring and evaluation of performance for mobile broadband networks in heterogeneous environments.

Acquiring access to this platform allows for the deployment of vast measurement setups to collect data from operational MBB networks in various European countries. Differently from other approaches based on operator-driven quality-assessment campaigns [26, 27], or on traditional drive-by tests [28], MONROE offers an open platform for repeatable and traceable experiments. Besides, it offers open access to collected data, which refer to multiple operators, and includes device-level metadata, which is the key to use and possibly filter results without raising user's privacy concerns. Therefore, this platform offers much richer data than what can be offered by crowdsourcing initiatives like, e.g., Netalyzer [29] and Haystack [30].

In this Chapter, we use the MONROE platform to investigate session-related performance statistics linked to the use of an off-the-shelf, WebRTC-based streaming application. This application enables streaming videos in real time with high quality using Web browsers that support WebRTC (e.g., Chrome, Firefox) [10]. When using Google Chrome, data from both the sending and receiving parties in a WebRTC-based *telemeeting*, can be gathered via the WebRTC internals page,[2] thus making it possible to get a complete overview of the stream in the mobile nodes. Such session-related statistics may help to identify root causes and track the origins of performance issues in video streaming, so as to understand how these technical factors impact Quality of Service (QoS) offered by the network and Quality of Experience (QoE) enjoied by the users. Indeed, gathering such insights is crucial and may steer the development of real-time communication schemes and intelligent optimization strategies. Our results show that current European MBB networks provide static users with enough resources and QoS to suitably make use of WebRTC, whereas mobility dramatically worsens network performance, often resulting in unacceptably low levels of QoE.

In the reminder of the Chapter, we first give an overview of WebRTC in Section 4.1. Subsequently, we present our measurement setup in Section 4.2. Section 4.3 focuses on experimental results. Eventually, Section 4.4 summarizes the findings of the Chapter.

4.1. WebRTC Overview

4.1.1. Real-time communications to and from browsers

The initial idea behind the development of a mechanism for Web-based real-time communication like WebRTC or other proposals discussed in standardization fora was mainly to provide a tool to empower Web browsers and make multimedia communications easier than before [10]. Specifically, the WebRTC approach is based on well-known Web technologies like HTML and JavaScript, which are able to simply integrate RTC into a

[2] chrome://webrtc-internals/

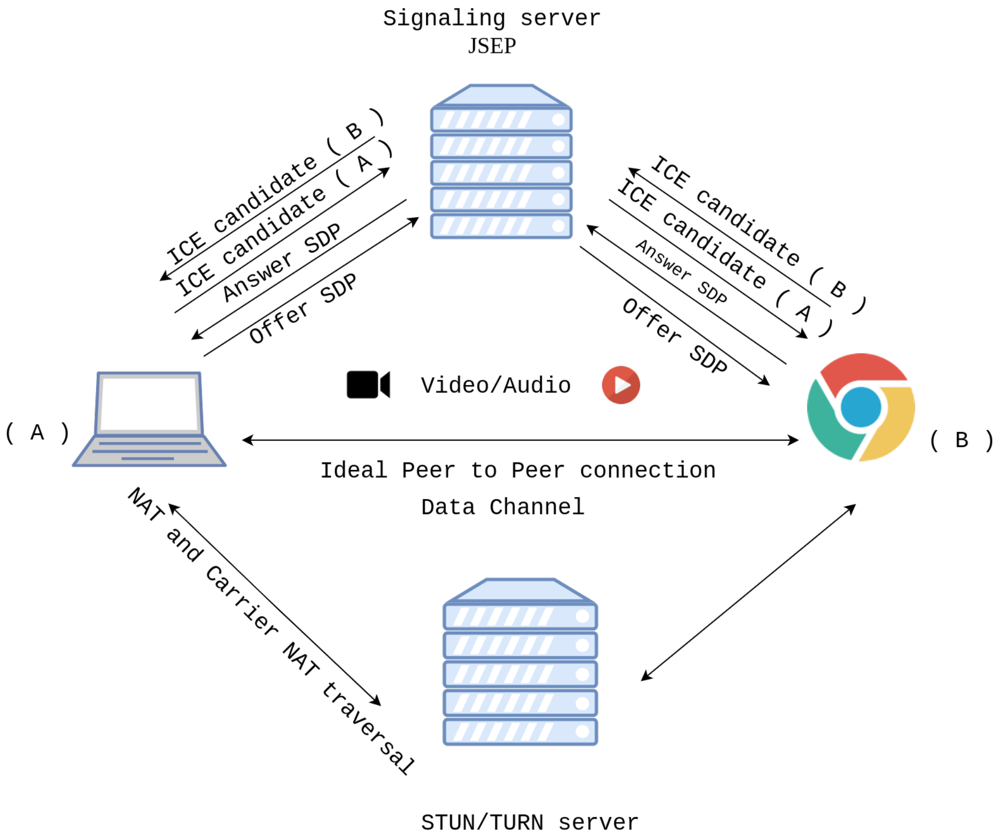


Figure 4.1: WebRTC peer-to-peer communication.

webpage.

Supported by Google first, and then by other Web browser developers (Firefox, Opera) and telecom vendors (Ericsson, Cisco, Alcatel-Lucent) [11], WebRTC integrates video streaming capabilities into Web browsers without the need of installing plugins or third-party software. Its standardization—led by W3C and IETF—helps separating application and communication level duties of video services [77]. In particular W3C and IETF are jointly concentrated on defining JavaScript Application Programming Interfaces (APIs) and a peer-to-peer communication mechanisms between browsers, to allow direct and possibly server-less video communication [78]. The designed API for WebRTC is capable of managing a browser based client-side RTC with host-to-browser connection, browser-to-browser connection management, encoding/decoding, NAT traversal and media streaming [78].

The main API highlights are as follow:

- **GetUserMedia**: it provides with agile access to user media such as microphones, cameras and display.
- **RTCDataChannel**: it authorizes data transfer through peer-to-peer channels.

- **RTCPeerConnection**: it is responsible for setting up a direct connections between two WebRTC applications, which allows data channels and media streams to be carried.

This API represents the base of any WebRTC application. Besides, it is possible to add plugins to exploit WebRTC or to empower it with, e.g., encryption and security features [79].

4.1.2. Protocols and Communication Services

Web browsers have to support several communication protocols and communication services to enable WebRTC. As shown in Figure 4.1, this includes mechanisms such as signaling, session establishment, transport and security.

For transport, WebRTC commonly uses UDP with DTLS security, which is a TLS extension for unreliable datagram transport. However, WebRTC gives the option to use TCP with TLS as well. In addition, it uses the Stream Control Transport Protocol (SCTP) and the Secure Real-Time Transport (SRTP) to control media channels and handle encryption keys. SCTP is capable of multiplexing multiple application data streams and provides reliable delivery of UDP datagrams. Hence, a peer-to-peer secured media path can be established by leveraging SCTP and SRTP.

For what concerns the session setup and the negotiation of media features and connection parameters, WebRTC uses SIP or XMPP with parameters conveyed through the Session Description Protocol (SDP). In addition, WebRTC uses the JavaScript Session Establishment Protocol (JSEP) as session setup mechanism, which endorses secured signaling, and encrypted data transport over either DTLS/UDP or TLS/TCP. It also uses signaling servers that help initiate peer-to-peer multimedia capabilities handshaking and establish connections.

WebRTC also includes mechanisms for firewall and NAT traversal. Specifically, it inherits from the VoIP world the Interactive Connectivity Establishment (ICE) framework. ICE helps to use TURN/STUN, which in turn eases WebRTC peers to test and collect preferred candidates of communication options, i.e., it sorts the known transport addresses of a media destination to which is possible to attempt connection. Indeed, carefully selected communication candidates are the key to maximize the chance of success.

The most recent specifications on WebRTC have been released in November 2017 with the W3C EditorDraft "WebRTC 1.0: Real-time Communication Between Browsers".[3]

[3] https://www.w3.org/TR/webrtc/

4.2. Measurement Setup & data-set

4.2.1. Setup

The main idea for the WebRTC experiment is to test video streaming using WebRTC for mobile users. Figure 4.2 outlines the streaming setup. When an experiment is scheduled on a set of MONROE nodes, each node runs the `WebStreamer` container, which makes an HTTPS link available to watch the video stream coming from the node. We use a Chrome browser acting as the WebRTC client in our lab. The WebRTC client is connected from a computer wired to a well provisioned gigabit link so that the receiver is not the bottleneck. The video stream produced by the MONROE node goes through a cellular uplink, traverses the Internet and then accesses the network of our lab. Therefore, the network bottleneck experienced in a WebRTC streaming session is the MBB network used by the MONROE node.

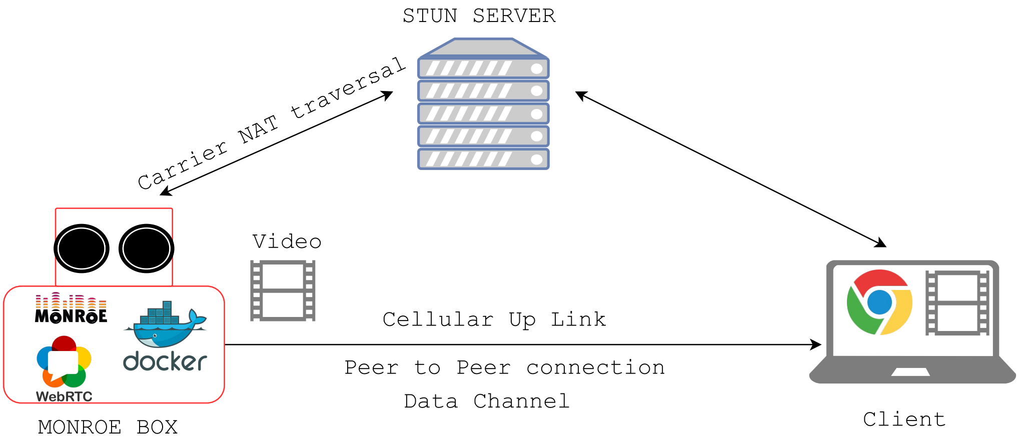


Figure 4.2: MONROE-WebRTC video streaming setup.

The video source is stored locally in the Docker container, with a resolution of 854x480 and a constant frame rate of 24 fps. The duration of the video is 9 minutes but can be looped for several hours, according to the duration of the scheduled experiment. The video streaming configuration parameters include the following:

- The possibility to use a local or online video by passing an RTSP link.
- Picking the preferred port for communication.
- Using a specific STUN server in case the client is behind a firewall.
- Tweaking resolution and duration of the video to stream.

At the client side, the video stream is received by means of Google Chrome, which also measures and collects streaming statistics in JSON format. The resulting dump contains peer-to-peer connection status information between the MONROE node and the

WebRTC client in our lab, in addition to updates and data statistics that can be easily accessed, e.g., by loading a specific Chrome page.[4]

The client logs contain, per each individual stream, the timing and headers of packets received as well as the timing of various internal events such as received frames, losses, bitrate, delay, jitter and other metrics that will be discussed in Section 4.2.2.

The MONROE platform provided us with many static and mobile nodes. Specifically, for the experiments reported in this Chapter, we have used the nodes listed in Table 4.1. The table includes some limited yet important information about the location of the node. Indeed, we only report operator name and country, whereas more detailed pieces of information about the locations are omitted since this study does not serve as a thorough survey of operator's performance comparison. The examples of results reported here are only meant to highlight how the potentials of WebRTC have not been fully unleashed by the sub-optimal MBB service available in many places in Europe as of today. However, as we will see later, MBB operators already offer WebRTC-ready connectivity for static users, while mobility is still a big issue.

The WebRTC throughput offered by MONROE nodes—i.e., the one offered by the MBB operator used at the node—differs a lot across space and operators, as well as across time, as shown in the third and fourth columns of Table 4.1. Such throughput values have been collected with capacity experiment run a few minutes before and after running the WebRTC experiments.

Table 4.1: MONROE nodes throughput comparison

Node ID	Location	Download [Mb/s]	Upload [Mb/s]
423	Telia SE	0.178-10.38	0.165-1.58
428	Telia SE	0.53-13.97	0.25-2.4
429	Telenor SE	0.78-12.178	0.22-1.74
501	Telenor SE	36.43-68.12	14.93-22.32
591	Vodafone ES	24.58-72.64	17.36-23.14
596	Vodafone IT	32.78-65.41	14.21-25.10

4.2.2. data-set

The WebRTC Docker container generates two types of log files. The first one is our Docker container log, which consists of the following:

- HTTP link to connect to.

```
2018-xx-xxTxx:xx:xx your url is: https://xxxx
```

[4] chrome://webrtc-internals

- The videos you are going to watch.

```
2018-xx-xxTxx:xx:xx.xx answer:[
2018-xx-xxTxx:xx:xx.xx "xx.mp4,rtsp://xxxxx"]
```

- The open port listening to upcoming connections.

```
2018-xx-xxTxx:xx:xx.xx HTTP Listen at :8888
```

- The constant Framerate of the video.

```
2018-xx-xxTxx:xx:xx.xx a=framerate:24.0
```

- The video codec used that includes h264, VP8, and VP9.

```
2018-xx-xxTxx:xx:xx Created a data sink for the "video/H264/VP8/
VP9" subsession
```

- The STUN server used when establishing connection between client and host.

```
  2018-xx-xxTxx:xx:xx.xx{"url" : "stun:stun.l.google.com:19302"
2018-xx-xxTxx:xx:xx.xx      }
```

- The preferred candidate for connection.

```
body:{"candidate":"candidate:0 2 UDP 2122252542
192.xx.x.x 42479 typ host",
"sdpMid":"sdparta_0","sdpMLineIndex":0}
answer:1
```

- The Video Content Type which is an extension used to communicate a video content type from sender to receiver of RTP video stream.

```
2018-xx-xxTxx:xx:xx.xx
a=rtpmap:96 mpeg4-generic/48000/2
```

When the connection is established, the WebRTC receiver starts collecting the JSON dump from Google Chrome, which is our second log. This includes different metrics: *bitrate received, frames decoded, packets lost during the session, packets received per second, video current delay in ms, frame rate received, decoded and output, jitter delay in ms, number of ACKs sent, target delay in ms.*

The MONROE-WebRTC template and the visualization tool that helps import all the metrics mentioned in CSV format can be found at [80] and [81].

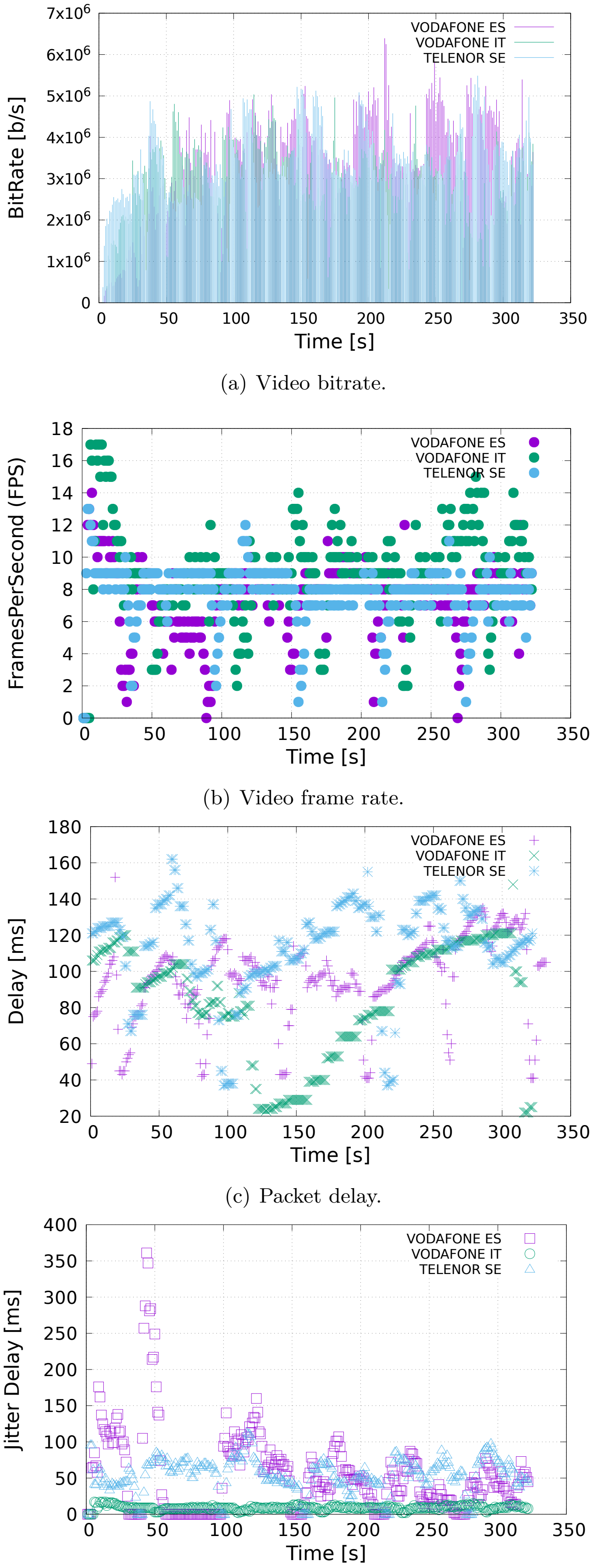
4.3. Results and Observations

In this section we report statistics on WebRTC performance attributes, as observed by means of Chrome internals at the destination of the WebRTC streams. We run WebRTC streamers in static nodes across Italy, Spain and Sweden, and on mobile nodes in Sweden. We only report a small yet indicative subset of the results we have collected. For each experiment, we plot the following statistics: (*i*) BitRate, (*ii*) frames per second (FPS) rendered at the receiver, (*iii*) packet delay and (*iv*) jitter. We compare static and mobile cases.

Static case. In the first scenario that we consider, the streamer and the client have high quality connectivity (nodes 501, 591, and 596 in Table 4.1). The selected MONROE nodes were connected to various 4G operators and, at the other end of the connection, we use a computer connected to a gigabit LAN directly connected to a multi-gigabit metropolitan network. In most of the test cases, the media stream turned out to be smooth, the final user having a rather good experience with typical bitrates of a few Mb/s, frame rates often above ten frames per second and latency in the ballpark of 100 ms, which is below the treshold of human perception. Jitter was normally below 100 ms. Figure 4.3 illustrates an example of performance figures over time for three simultaneous connections using MONROE nodes in three countries. Of course each connection shows different performance figures, and in particular the Spanish sample shows the worst behavior in terms of rate and delay/jitter, although we need to mention that the selected MONROE node in Spain was using an Italian SIM card operating in roaming. In any case, the observed performance is acceptable.

We show sample results for multiple operators in four countries in Figure 4.4, in terms of bitrate and delay. In there, we see that the media stream was smooth in most of the cases, with limited delay (and delay variations, i.e., jitter), and bitrates of a the order of a few Mb/s, corresponding to acceptable performance.

However, the results for Swedish operators are not very good, which is in contrast with other observations on the quality offered by those operators. This is an example of experiment that needs to be interpreted jointly with metadata. In fact, observing our logs, we have discovered that the SIM cards used for static WebRTC experiments in Sweden had simply exhausted their monthly data allowance, which resulted in severe rate limiting experienced by the MONROE nodes, and low WebRTC bitrate.



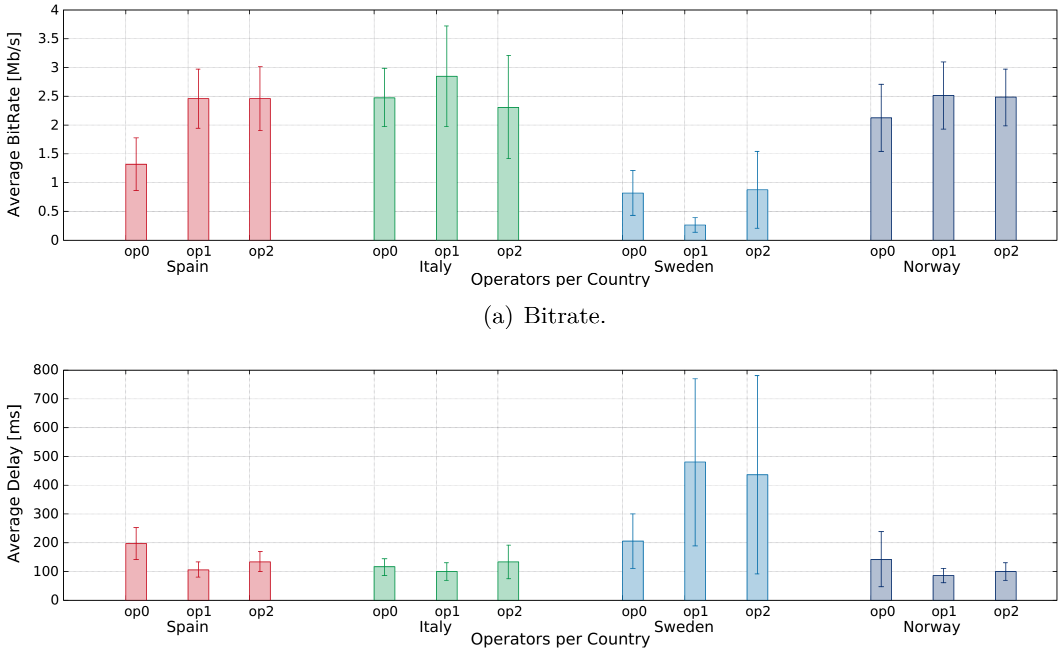
(a) Video bitrate.

(b) Video frame rate.

(c) Packet delay.

(d) Jitter delay.

Figure 4.3: WebRTC performance experienced by static nodes under different operators in different countries.



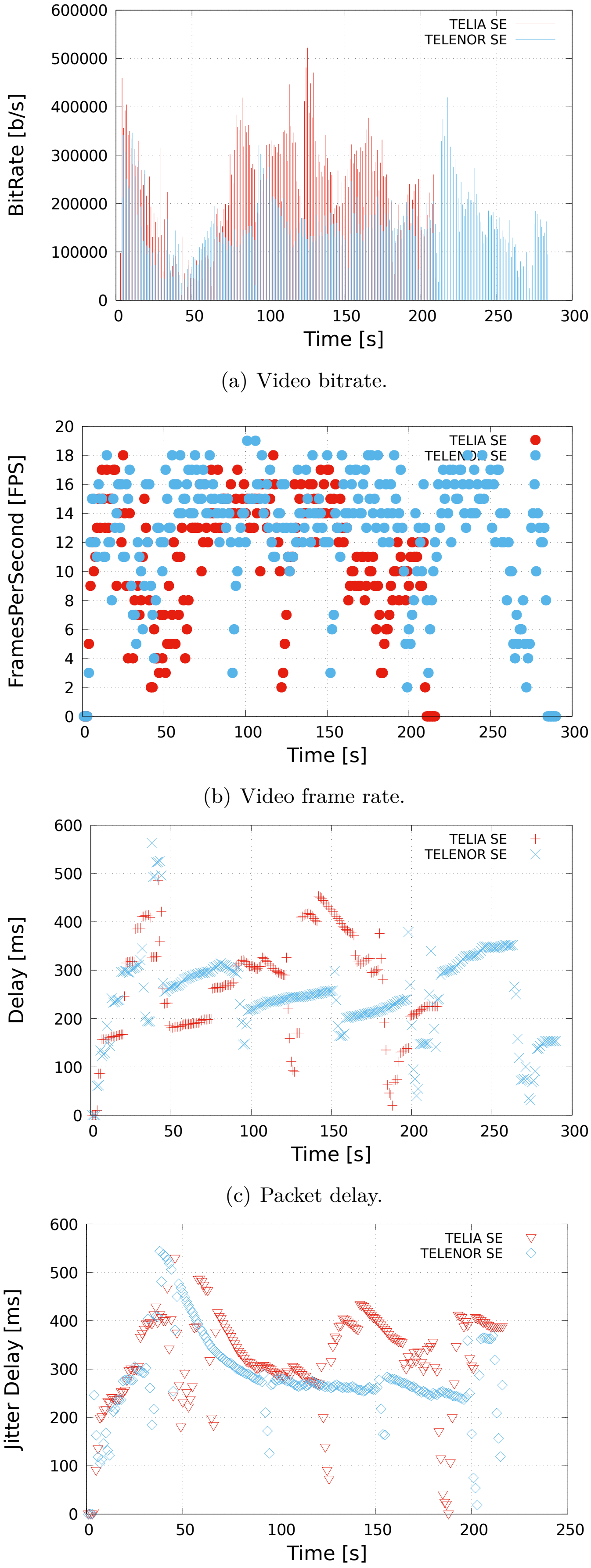
(a) Bitrate.

(b) Delay.

Figure 4.4: WebRTC performance figures observed for static nodes.

Mobile case. In this scenario we select MONROE nodes in buses in Sweden (nodes 423, 428, and 429 in Table 4.1), while the other end of the connection is in our lab, as in the other case. Figure 4.5 shows that the user experience was not as smooth as in the first case since the bitrate was significantly degraded. Although WebRTC was able to keep the stream going, frame rate was very variable, and delay and jitter were annoyingly high in all locations and for all operators. We have observed many other cases like this in our measurements (and not only in Sweden, of course), which leads to the conclusion that current MBB networks are not ready to fully support WebRTC (and alike multimedia services) on the move.

To further understand the behavior of WebRTC on MBB networks, Figure 4.6 shows 3D plots of bitrate and delay experienced by a MONROE node on a bus, for one of the operators in Sweden, as a function of the geographical position of the node (also shown in the topmost subfigure using Google Earth). The information about coordinates is provided by the MONROE node itself, within a metadata stream generated during the experiments. From the figure it is clear that some areas crossed by the bus were very poorly served, so that delay are constantly high and bitrates change smoothly over the trajectory, indicating that coverage (signal strength) is far from optimal in that area.



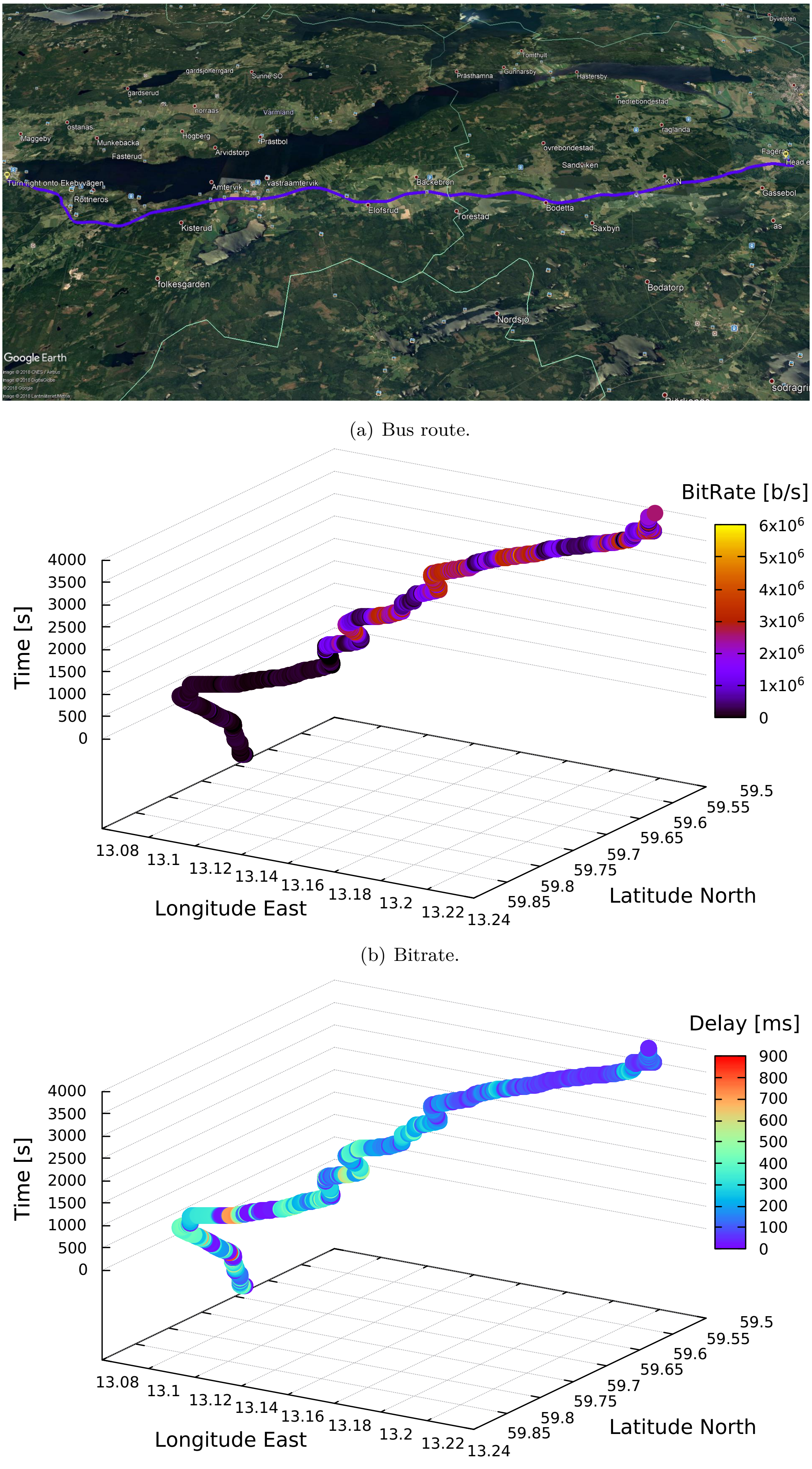
(a) Video bitrate.

(b) Video frame rate.

(c) Packet delay.

(d) Jitter delay.

Figure 4.5: WebRTC performance figures observed for a bus in a medium-size city in Sweden.



(a) Bus route.

(b) Bitrate.

(c) Packet delay.

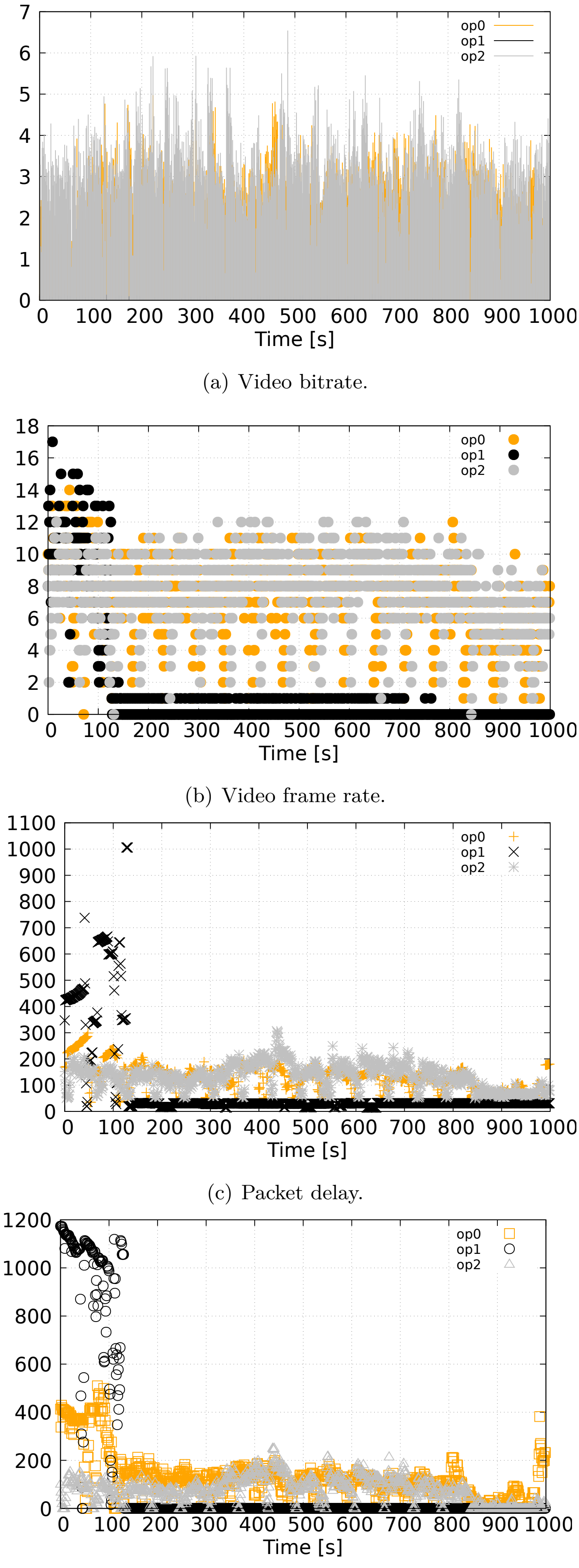
Figure 4.6: WebRTC video performance measured at destination, on a public bus on service in a medium-size city in Sweden.

Figure 4.7 depicts instantaneous values of bitrate received, video frame rate reconstructed at destination, and video delay and jitter experienced by the receiver. The figure shows that performance under mobility can be relatively good, at least under good cellular coverage. However, cellular coverage in mobility is not always available in the observed area. By analyzing the metadata associated with the traces reported in Figure 4.7, we have noticed that one of the trains used for the experiment progressively moved from the city center to the outskirts of Oslo, while the other trains stayed within the city, moving between multiple stations in Oslo. The streams from the trains remaining in the city experienced almost always good performance, albeit with high variability. On the other hand, the stream from the train moving outside of Oslo shows poor performance at the beginning of the experiment, when the stream suffered a huge degradation in bitrate and a significant increase in delay and jitter, and later suffers an almost complete connection drop (almost no delay reports were received, and only sporadic frames were reproduced).

From the above results, it is clear that peer-to-peer communication over the Internet is still not *ideal* and whether one gets good QoE or not for the whole communication period completely depends on how good the network connection is, and how good it remains across the whole time of the communication session. In the ideal WebRTC scenario, endpoints are Web browsers running on reasonably powerful laptops with strong WiFi or wired network connections, communicating on top of a reasonably consistent network. This should work well. However, if the devices are mobile and have a non-consistent and often not good wireless connection, then the quality of the communication is likely to be below any acceptable treshold, as observed in the mobile case described above.

4.4. Discussion

In this Chapter, we have evaluated the performance of WebRTC for static and mobile users by leveraging the MONROE platform. To this aim, we have designed an open tool, running in a Docker container, for generating WebRTC sessions in mobile nodes by using standard WebRTC APIs. As such, the work presented a complete and novel methodology for the performance evaluation of Web services using operational MBB networks. As an initial result, we have observed that mobility is still an important challenge for WebRTC, since MBB operators do not yet provide users with full quality coverage on the move. Our approach can be used in the future for a broader and continuous assessment of WebRTC.



(a) Video bitrate.

(b) Video frame rate.

(c) Packet delay.

(d) Jitter delay.

Figure 4.7: WebRTC performance figures observed for a train in a medium-size city in Norway.

5 Performance Evaluation of QUIC

HTTP prevails as the undisputed king of the Web with its two versions HTTP/1.1 and HTTP/2 adopted by the vast majority of websites. This is also clear in terms of Internet traffic [82] and in terms of the widespread support offered by all current browsers to HTTP/1.1 and HTTP/2, based on TCP and all its features. However, in recent years a new protocol has come into play with the name of QUIC (Quick UDP Internet Connections) as the foundation of upcoming HTTP3 [39]. QUIC is a new network transport protocol originally designed and proposed by Google, which operates on top of a simpler transport protocol, UDP, so as to bypass the limitations of legacy TCP/IP transport. QUIC itself is commonly seen as a transport-layer protocol, although it embeds encryption features and relies on conventional UDP transport.

QUIC eliminates significant TCP bottlenecks such as the need of initial TCP handshake mechanism, which takes one Round Time Trip (RTT) or two, in case of TCP data encryption via TLS. Indeed, Google has designed and implemented a novel encryption scheme for QUIC, similar to TLS, which couples connection establishment and key agreement within one RTT. Nevertheless, QUIC can start a connection in zero RTT, like UDP, by instantly sending encrypted application data to the server. This is possible when it previously has cached in a server certificate from a previous connection. Moreover, running on top of UDP, QUIC bypasses HoL issue of TCP in case of packet loss. For these reasons, in the long run, QUIC is expected to replace TCP and TLS in the Web [13].

Since offering video services and multimedia channels is the main trend in the Internet, which is stressful for the entire network infrastructure and in particular for MBB networks, the evaluation of QUIC and HTTP/3 as an alternative to TCP and HTTP/1.1 or HTTP/2 is necessary in the steadily growing MBB environment [24]. This need motivates us to design, develop, and deploy experiments with QUIC and HTTP/3 on operational MBB networks.

Thus, we focus on QUIC and HTTP/3 performance figures in mobile environments, for which so far little experimental work exists. In fact, other works on assessing QUIC performance figures are currently growing, but only by studying basic properties in controlled environments. For instance, the authors of [16] use Google's server and the client to evaluate the performance of QUIC and TCP in wireless networks in a controlled environment although they have not tested the service offered by mobile operators and core networks. Similarly, the authors of [17] have QUIC and TCP in the kernel level through a gigabit LAN environment.

To assess QUIC and HTTP/3 in uncontrolled networks, we need, on the one side, an open and flexible implementation of QUIC, which allows for end-to-end testing and collecting statistics, possibly playing with QUIC's configuration. We use two open-source implementation of QUIC for this purpose: `flowsim`[1] and `Mvfst`[2]. We also resource to *qlog files*, to extract statistics [23]. On the other side, we also need objective information about MBB performance and reliability to provide an adequate QoE (Quality of Experience) to the end-user when using QUIC. Various initiatives have arisen recently to help in that direction, among which the above mentioned US FCC's Measuring Broadband America initiative and the European initiative MONROE. In our work, we leverage MONROE, to which we have direct access and which we have co-designed during the last years. Our results show that QUIC and TCP performance attributes over MBB are relatively similar, except QUIC reacts better to the vagaries of channel quality experienced by mobile nodes. However, with the application protocols, HTTP3 based on QUIC performs better than HTTP/1.1 and HTTP/2 based on TCP, although the choice of the QUIC's control algorithm sensibly impacts on the overall performance.

The main contributions of this Chapter can be summarized as follows: (*i*) we design Docker containers that run QUIC and TCP measurements as well as run HTTP variants, and integrate them in the MONROE platform; (*ii*) we explore the performance of QUIC and HTTP/3 to download contents from the Web under heterogeneous conditions, in terms of geographical diversity and mobility; (*iii*) thanks to the flexibility of `flowsim`, which can act as a TCP or QUIC traffic generator as well as an HTTP server, we compare the performance of QUIC, TCP and compare HTTP variants; (*iv*) thanks to the flexibility of `Mvfst`, which has been developed by Facebook, we evaluate the impact of congestion control algorithms in QUIC; and (*v*) we generate a large open *qlog* data-set with our experiments with QUIC and HTTP/3 in real operational networks.

The rest of the Chapter is organized as follows. Section 5.1 offers background information on QUIC and the *qlog* file structure. Section 5.2 discusses our measurement methodology and setup. Section 5.3 focuses on experimental results. Eventually, Section 5.4 summarizes the findings of the Chapter and points out future research directions.

[1] https://github.com/paaguti/flowsim

[2] https://github.com/facebookincubator/mvfst

5.1. QUIC operation and logging

5.1.1. The QUIC protocol in a nutshell

QUIC is a new evolving transport protocol originally proposed by Google to compete specific problems encountered while using TCP's HTTP/1.1 and HTTP2. QUIC provides similar guarantees to TCP in terms of reliable data delivery and strict flow control by combining TCP multiplexing and TLS into one transport protocol built on top of UDP [39].

To achieve that, QUIC employs features such as Multiplexed Streams [83], which eliminates the HoL problem found in TCP by extricating the ordering restrictions on packet delivery since in-order data delivery is done on the stream level rather than the connection level.

A key feature of QUIC is the shorter time to establish a secure connection within fewer RTTs than TCP/TLS. To achieve this, the TLS negotiation mechanism is part of the QUIC protocol by default, removing the need for a separate non-TLS TCP handshake to synchronize client and server [84].

These new features pave the way to implement HTTP/3 [85] as the upcoming web application protocol. HTTP/3 will evolve radically by avoiding the use of HTTP/2's stream metadata, by using a new header compression algorithm as well as by adopting a new prioritization scheme.

Another interesting feature of QUIC is that it can be rapidly deployed since it runs in the user space and not in the kernel space of the operating system. This allows researchers and developers to quickly test and deploy and evaluate flow-control and congestion-control alternatives.

As for the loss and recovery mechanism, which is run on top of connectionless UDP features, QUIC introduces monotonically-increasing packet numbers to differentiate between new and retransmitted data, which significantly simplifies RTT measurements [84].

However, QUIC's diversification of features is a double-edged sword that might contribute to generate performance inconsistencies across implementations. This raises the need to investigate QUIC's performance under different real cellular networking environments, with different congestion control algorithms.

5.1.2. QUIC logging

QUIC is characterized by a series of messages which govern the end-to-end interaction between a server and a client. As such, QUIC counts with a predefined set of events which can be captured and analyzed offline. Well-established protocols like TCP rely on specific analysis and debugging tools that directly present the protocol information as it would be seen externally by another entity. Existing tools for TCP rely on pcap files, which simply capture IP packets (or even lower layer frames) and read into protocol headers to reconstruct TCP flows and events. This approach, however, prevents the access to state variables that are internal to the communication and are not conveyed through protocol headers, although they can help the inspection of more intricate elements, such as congestion control algorithms. Thus, differently from the case of TCP, debugging a protocol as complex as QUIC, which is not a legacy transport protocol and comes with native encryption, is not feasible through the collection of legacy pcap files [23].

To this aim, the QUIClog project [23] was created to define a specific format, *qlog*, to organize and record, in so-called *qlog* files, different information captured during an exchange over QUIC. Logging occurs within the QUIC implementation, at server side. However, it must be said that several implementations of HTTP/3, which runs over QUIC at an application level, have managed to capture logs on the client side as well. As our study will center on the evaluation of QUIC's transport-based capabilities, this possible scenario will not be explored in the Chapter. Although QUIC developers are free to introduce additional information in *qlog* [86], conferring to the log a certain degree of extensibility, some *qlog* fields are reserved and predefined, and a general basic indexing structure must be guaranteed to parse information correctly.

A *qlog* file describes protocol events, not just QUIC events, according to the scheme shown in Figure 5.2. Each event record must include, as specified in [87], a timestamp indicating the time at which the event was registered, a field describing the type of event associated to the record, and finally, the data registered. Events can be grouped in categories, which although not mandatory are recommended to favour high-level filtering of logs. It is also essential to note that a *qlog* can trace activity from different protocol stacks, thanks to the fact that a field is reserved to indicate the type of protocol the event is related to, making it possible to capture both TCP and QUIC information.

Regarding event definitions related to QUIC traces, using *qlog* allows to record information related to the connection and key-sharing processes, as well as the data-exchange process. It does so not only gathering information about the frames sent and received from the serve-side, but also by including transport metrics relevant for the communication, such as RTT and congestion window updates. As a matter of fact, these values, specially time delays, are aggregated and classified under the recovery category, whose event types are specified in a generic way to ensure that diverse congestion and recovery schemes can be covered by the logging format. It is mainly thanks to the information gathered in the recovery category that it is possible not only to evaluate QUIC's performance over time but also to fine-tune the protocol, as congestion and loss detection parametrization is also recorded in *qlog* files. To improve the interpretability of results, a visualization tool, `qvis`,[3] is also available for users. As pictured in Figure 5.3,

[3] https://qvis.quictools.info

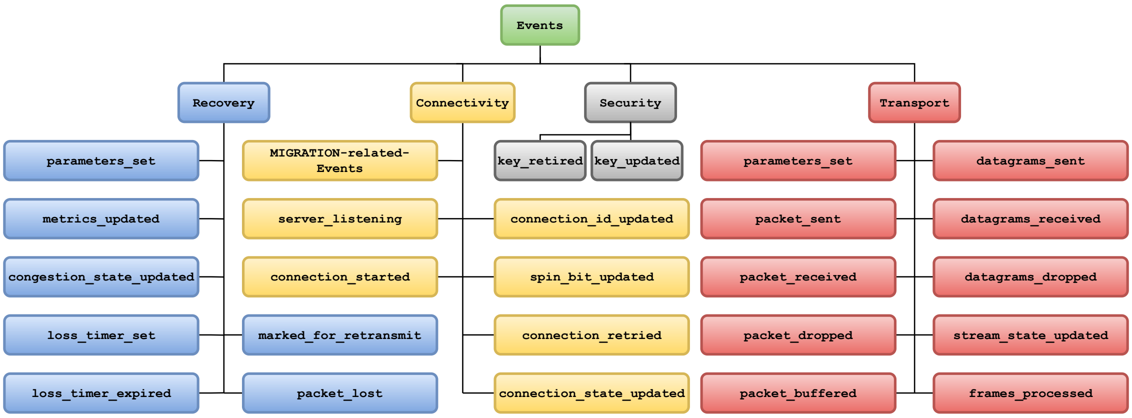


Figure 5.2: QUIC events used in *qlog*

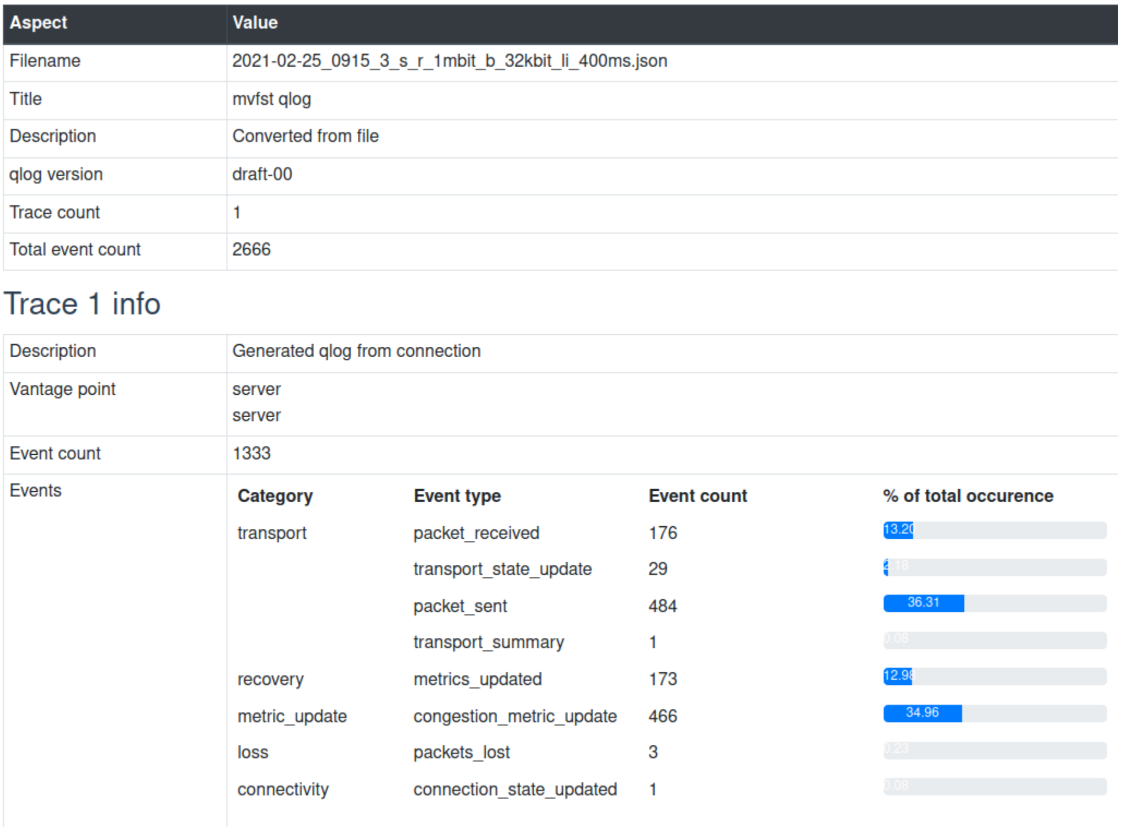
qvis allows for the import of logging files and the graphical display not only of the packet exchange between nodes but also of aggregated statistics that can be extracted from *qlog* events, such as packet loss percentage, as well as event count and percentage of occurrence. For all the aforementioned, *qlog* files are currently the main source of insight to understand the behaviour of QUIC.

5.2. Experimental Methodology

The QUIC protocol has been evaluated mainly in controlled and simulated scenarios, which is useful to understand the basic behavior of the protocol. Here, we want to capture the complexity of operational cellular networks, where radio conditions impact performance and can vary immensely in a brief period. As such, we want to shed light on the behavior of QUIC in the context of realistic mobile scenarios. Moreover, aware of the difficulties for researchers to count with a testing environment that allows them to experiment over operational networks and use existing infrastructure, we wanted to contribute by building and releasing a *qlog* data-set with the files extracted from our tests.[4]

In addition, we take into consideration that multiple open-source versions of the QUIC protocol are available, each of them offering distinctive tuning possibilities for the configuration of QUIC instances. This opens the possibility of tackling the comparison between the most well-known and widely adopted implementations, currently those being flowsim and Mvfst. It also opens the possibility to test the congestion control algorithms currently under evaluation at IETF for QUIC, which also entails the need to compare QUIC to TCP.

[4] data-set available at https://doi.org/10.5281/zenodo.4602217



Aspect	Value
Filename	2021-02-25_0915_3_s_r_1mbit_b_32kbit_li_400ms.json
Title	mvfst qlog
Description	Converted from file
qlog version	draft-00
Trace count	1
Total event count	2666

Trace 1 info

Description	Generated qlog from connection
Vantage point	server server
Event count	1333
Events	

Category	Event type	Event count	% of total occurence
transport	packet_received	176	13.20
	transport_state_update	29	2.18
	packet_sent	484	36.31
	transport_summary	1	0.08
recovery	metrics_updated	173	12.98
metric_update	congestion_metric_update	466	34.96
loss	packets_lost	3	0.23
connectivity	connection_state_updated	1	0.08

Figure 5.3: Example of qvis statistics view

5.2.1. Methodology

We study the case of downloads from a network server to a mobile node and the case of webpage retrieval with HTTP variants. The methodology chosen to assess experimentally QUIC's configurations follows an *outside-in* approach, where we first center on external information that can be extracted from the experiment, such as its duration or the throughput of the download. To do so, we frame the experiments in the context of cellular networks, and develop Docker containers compatible with the MONROE platform. Experiments are defined in the form of Docker containers embedding `flowsim` and `Mvfst` clients. The use of containers favours an efficient use of the MONROE node's resources and ensures that the experiment execution does not compromise the configuration of the node or future experiment runs.

We test and certify our containers with the MONROE authority, and obtain access to the MONROE automatic experiment scheduler, through which we deploy recurrent experiments at client side. Specifically, the scheduler deploys our containers with our transport clients on selected MONROE nodes when the time to run the experiment comes.

The server, which acts as a traffic generator and an event logger, runs in a lab machine in which we implement `flowsim` and `Mvfst` and tune the structure of *qlog* files to collect the desired flow statistics of both QUIC and TCP downloads. Figure 5.4 shows our experimental scenario, which includes the tools needs for storage and retrieval of measurements.

Since QUIC allows multiple parallel flows, and the fact that the selected tools can be run in multiple instance on the servers, and hence allow to receive multiple incoming requests in parallel with different server configurations, entail the consideration of two different types of download experiments: one in which the client issues download requests sequentially, and thus does not have to share network resources between flows, and a second one, in which multiple download requests are triggered in parallel from the client against different ports in the server (which corresponds to one client connected to multiple servers on the same machine). With this, we can test multiple configurations on QUIC in parallel, and also test their fairness. In addition, `flowsim` offers the possibility to mimic HTTP (versions 1.1, 2 and 3), so that we use it to reproduce the behavior of webpage retrieval with TCP and QUIC.

We repeat all types of experiments sequentially from multiple MONROE nodes and from each of the cellular interfaces available on the nodes (note that each MONROE node has three interfaces connected to three different MBB operators).

Although download/page retrieval time and throughput (or TCP goodput) measurements can provide an overview of the experiment's performance, it does not explain the reasons behind the results gathered, due to the complexity of the QUIC and TCP protocols. We then first analyze the logs of our Docker containers for aggregate download statistics, and then move to the analysis of *qlog* files redacted by our experimental server, which allows to evaluate the behavior of the transport protocol during the download.

Indeed, the inspection of experiment results and *qlog* files allows us to study the evolution of transport-related metrics. From the user point of observation, in the Docker container, we collect flow statistics such as the (webpage) download time, which is equivalent to estimate the average throughput during the download. Instead, with the *qlog* files generated at the server, we study the behavior over time of experienced RTT and throughput, and how the congestion window evolves and drives the volume and speed of data downloaded.

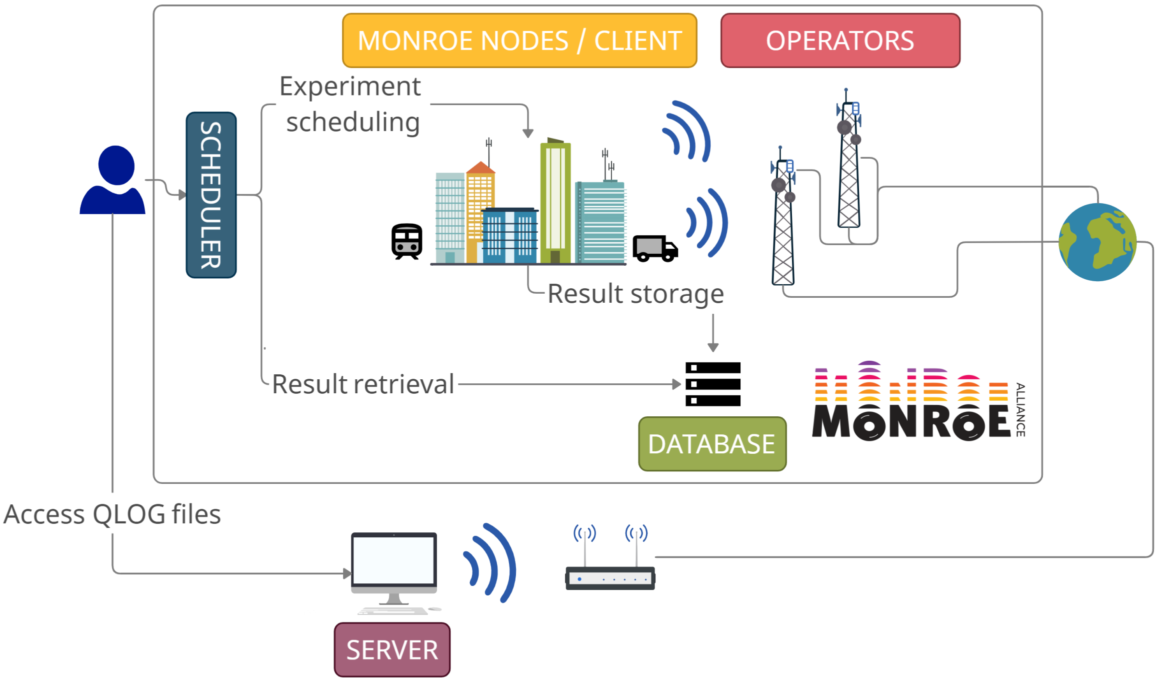


Figure 5.4: Experimental scenario

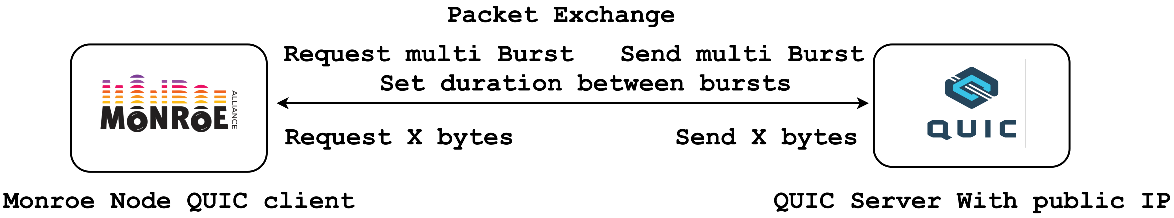


Figure 5.5: `flowsim` MONROE platform setup

5.2.2. Setup

Our Docker containers come with a QUIC/TCP client request file download from a server located in our lab that runs a QUIC/TCP traffic generator, as shown in Figure 5.4.[5]

A client in the container interacts with our server as exemplified in Figure 5.4 for the case of QUIC with `flowsim`. We have opened a limited number of ports on a few machines in our lab and across Europe in labs of the partners of the MONROE Alliance. All servers have public IP address, so that a client on a MONROE node, and hence passing through a cellular network and the public Internet, can request multiple bursts of traffic (i.e., file downloads).

We use clients and servers in Spain, Italy, Norway and Sweden. For the clients, we distinguish between static nodes and mobile nodes, the latter being MONROE nodes installed on trains in Norway and busses in Sweden. Static nodes have high-quality connectivity, while mobile nodes can suffer low quality and even outages. In all cases, the server-side of our experiment is connected to a gigabit LAN directly connected, in turn, to a multi-gigabit metropolitan network.

The Docker container at the client requests 100 kB to 1 MB per download burst in the experiments with `flowsim`, to test Web downloads, and set a time interval of 15 seconds for experiments with `Mvfst`, to emulate streaming cases. We repeat the download 10 times in each experiments from each node, alternating downloads to pause times during which we select a different cellular interface. The pause time is set to 10 seconds, and a data burst is interrupted after 60 seconds in case the file download or the page retrieval is not completed before. This is needed in order to avoid hanging connections during the experiments. We collect the download time in seconds for all requests as well as the underling *qlog* events.

In the experiments with `Mvfst`, we configure the download of each file several times, using a different congestion control algorithm every time, and alternating between single downloads and multiple parallel downloads. The Docker container with `Mvfst` uses `tperf` to tell the QUIC server which congestion control protocol has to be used. Indeed, the fact that QUIC runs in user space, makes it possible to reconfigure the server on the fly. We experiment with Cubic, BBR, Copa and Newreno congestion control algorithms, which are encoded in `Mvfst`. Instead, `flowsim` only implements Cubic.

For what concerns Copa, a fairly new congestion control scheme that follows the delay-based approach undertaken by BBR, we remark that we have not played with its optimization, and we have used a default Copa's latency factor of 0.5, which corresponds to an equilibrium between queueing delay and throughput [88].

[5] Our Docker containers are publicly available for interested experimenters at https://hub.Docker.com/repository/Docker/07777/fullquic and https://hub.Docker.com/r/7466291/zmq-mvfst-tperf

5.3. Results

We start first by comparing pure TCP with pure QUIC, followed by comparing application protocols HTTP (i.e., HTTP/1.1), HTTPS (i.e., HTTP/2), and HTTP3 (i.e., `flowsim`'s implementation of HTTP3 with QUIC).

We only report a small yet characteristic subset of the results we have collected and start first by showcasing the results gathered with `flowsim` and then move to `Mvfst`. For the interested experimenters, all the results are accessible online.[6]

5.3.1. Assessment of QUIC and HTTP/3 performance

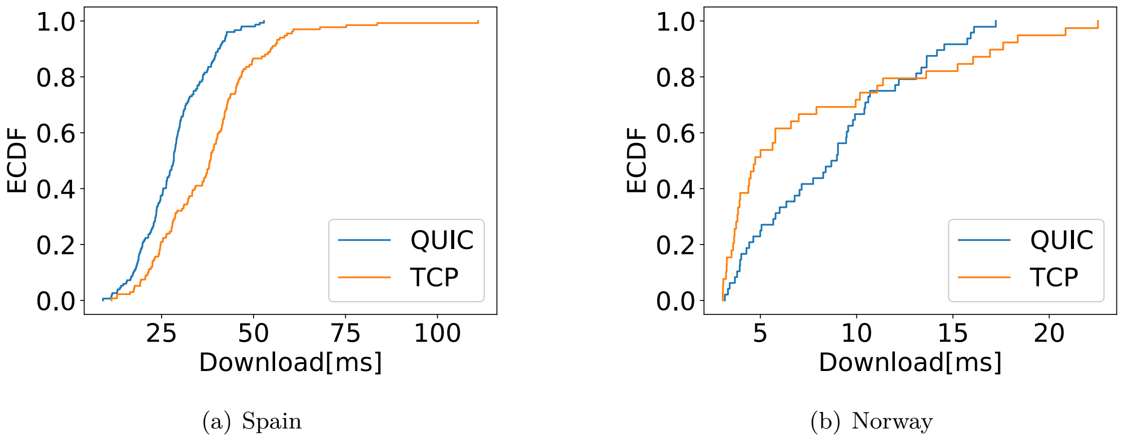
Here we report the results of our `flowsim` experiments for the evaluation of QUIC and HTTP/3 vs TCP and older HTTP versions, using a single stream per download.

Figure 5.6 and Figure 5.7 show that empirical distribution of the download time for traffic burst of 1 MB. The considered experiments were run from static nodes in Spain, Norway and Sweden. Figure 5.6 contains separated statistics for the three countries. Differences across countries are normal also because we put together the results of three operators per country, each with different latency to the servers. However, for each country we use a local server in that same country. The figure also reports TCP statistics for the same kind of experiment. It is interesting to see that QUIC performs better than TCP in Spain, but not in Norway and Sweden. The differences are however not huge.

Figure 5.7 reports download time statistics, and the collected samples, for the case of mobile clients located on busses in Sweden. The distribution of results shows practically no differences between QUIC and TCP. However, TCP tends to suffer occasionally and reports higher extreme values. This means that QUIC reacts better to channel outages than TCP.

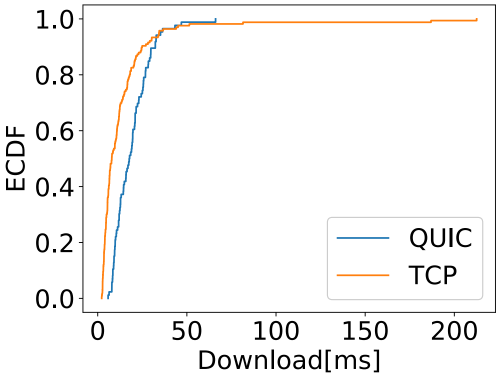
Next, we turn our attention to the application protocols, using the HTTP emulation mode of `flowsim`, and we change the download size to 100 KB to match the size of real webpages. We can notice a difference in download time, as observed in Figure 5.8 for Spain (with static clients) and Sweden (with clients on the move). In this case, HTTP3, which is based on QUIC, offers a clear advantage, especially with respect to HTTP/2, due to the reduced connection establishment time.

[6] https://github.com/Mohmoulay/QUIC-MedNetCom



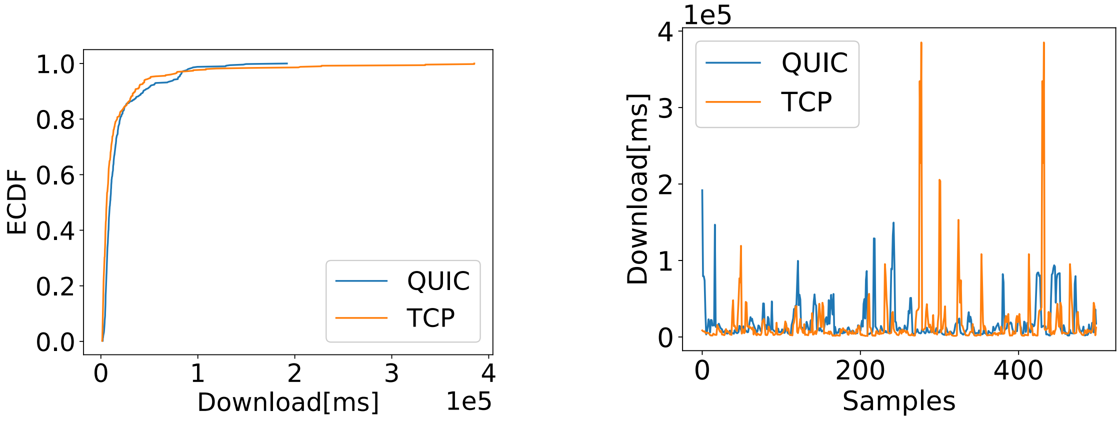
(a) Spain

(b) Norway



(c) Sweden

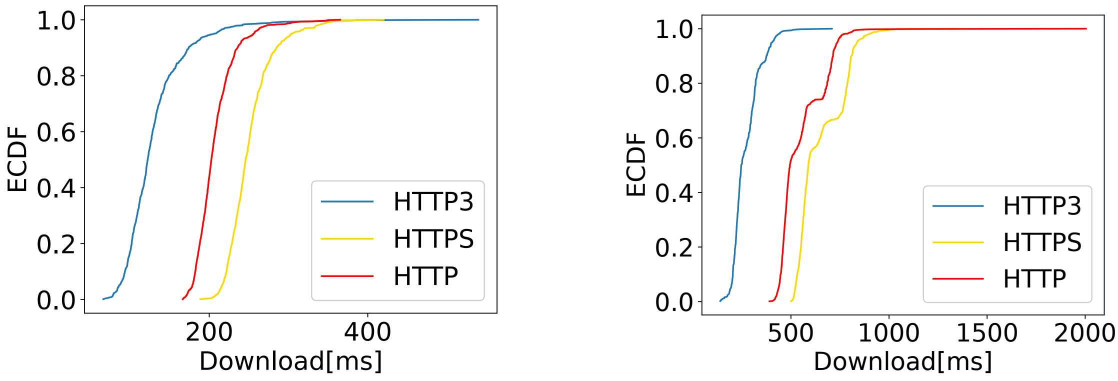
Figure 5.6: Download time comparison between QUIC and TCP, with `flowsim` clients in Spain, Sweden and Norway (download size of 1 MB)



(a) ECDF

(b) Samples

Figure 5.7: Download time comparison of QUIC and TCP with `flowsim` on a public bus in Sweden (1 MB downloads)



(a) Spain

(b) Sweden

Figure 5.8: Page download time comparison between HTTP versions with `flowsim` clients in Spain (static) and Sweden (mobile), with page size of 100 kB

We also collect *qlog* files for plain QUIC and HTTP3 experiments. Here we show the case of the mobile Swedish nodes. In Figure 5.9, we report RTT, throughput, congestion window and number of bytes in flight as reported in the *qlog* files for a specific experiment lasting about one second. From the time series Figure 5.9, we can see how the `flowsim` implementation of QUIC adapts fast to delay variations (due to the mobility of the client), and the resulting volume of bytes in flight is maintained practically constant after a short transient phase.

Similarly, for the case of HTTP/3 with a mobile client, Figure 5.10 shows that a QUIC-based Web browsing is quite robust to fast channel quality variations.

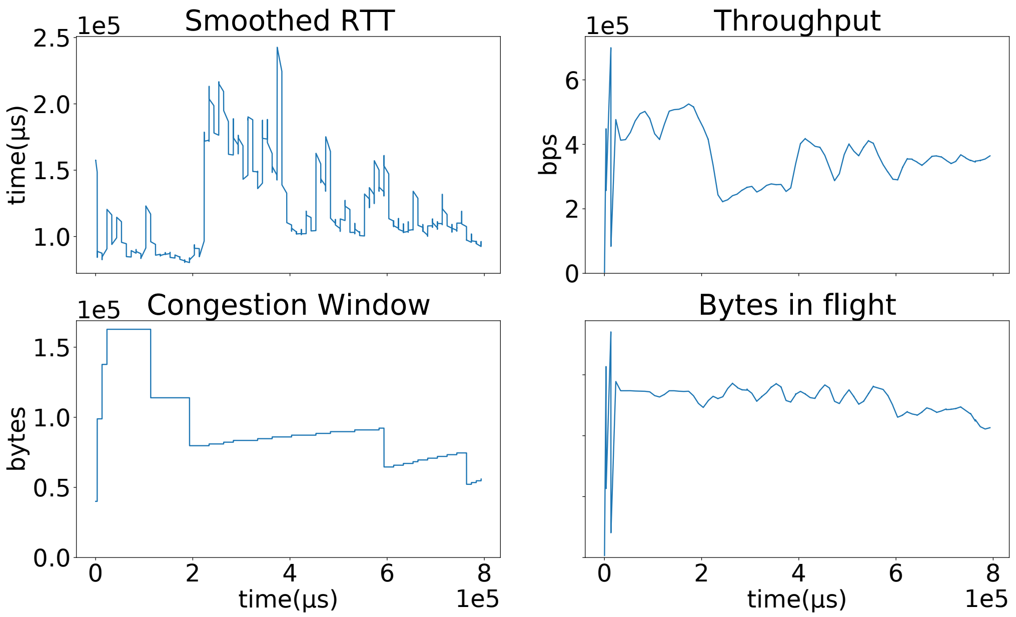


Figure 5.9: Time series for for a QUIC download from a mobile `flowsim` client in Sweden (*qlog* events, 100 kB downloads)

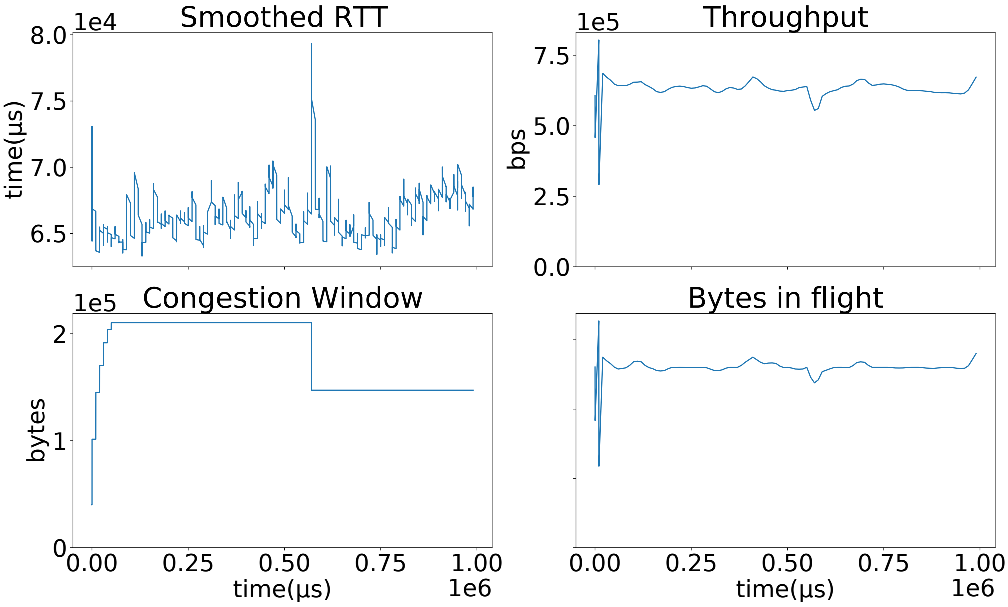


Figure 5.10: Time series for a webpage download from a mobile `flowsim` client in Sweden (*qlog* events, 100 kB downloads)

5.3.2. Assessment of Congestion Control Variants in QUIC

Here we evaluate the impact of congestion control on QUIC and how it behaves over time in the presence of multiple downloads streams. We use `Mvfst` for the set of experiments described in this subsection, because it allows to test several congestion control schemes. Results with `Mvfst` running Cubic are very similar to the ones obtained with `flowsim`, so we do not include `flowsim` experiments here. Instead, we compare sequential vs parallel experiments: in the sequential ones, the congestion control scheme is switched from one set of experiments to the next, while in parallel experiments, the various congestion control schemes are run in parallel on different instances of the QUIC servers, using different ports. This also allows us to evaluate the fairness among QUIC variants.

Figs. 5.11 and 5.12 report the results for a download stream, performed sequentially or in parallel, respectively, from the same static MONROE client in Spain. The figures include statistics extracted from the *qlog* files, and refer to the case of three streams per download (mimicking the download of multiple media channels), although results with a different number of streams are similar. The figures also report results for four congestion control schemes, the first (Cubic) being the one also used in the experiments with `flowsim`.

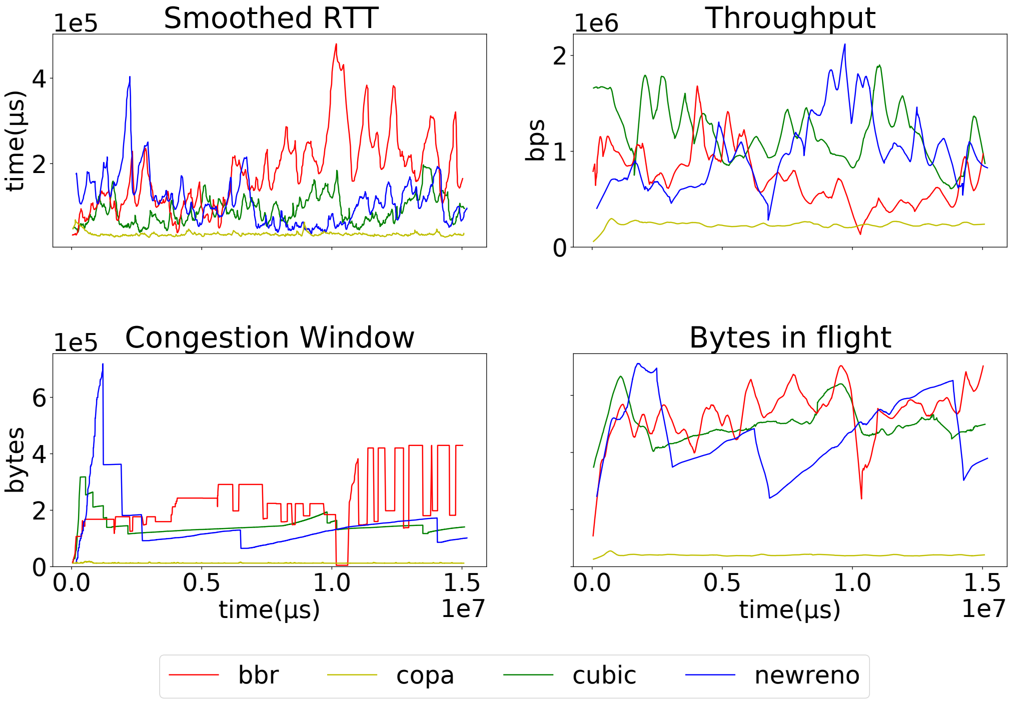


Figure 5.11: Sequential time series in Spain with three download streams (*qlog* files of `Mvfst` experiments)

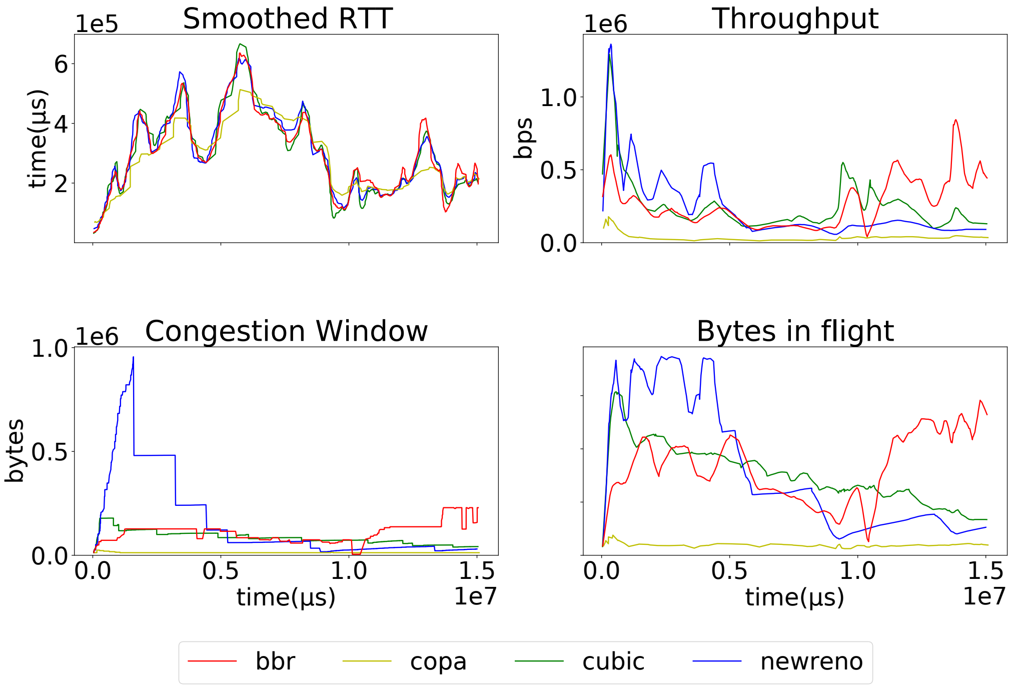


Figure 5.12: Parallel time series in Spain with three download streams (*qlog* files of `Mvfst` experiments)

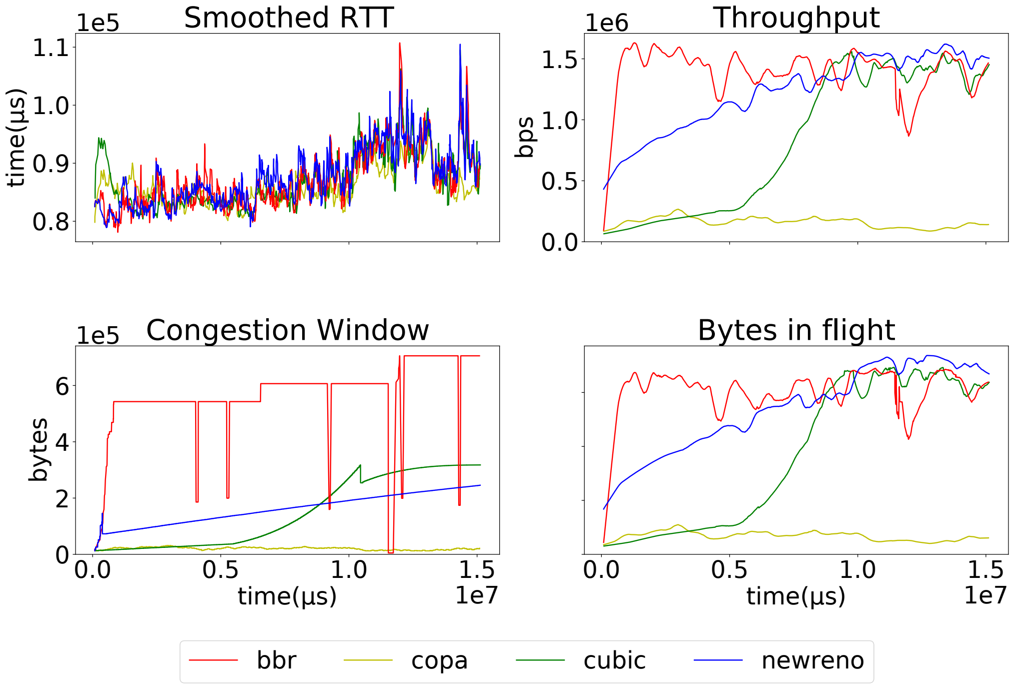


Figure 5.13: Parallel time series in Norway (with a static `Mvfst` client) with three download streams

In the *sequential* case of Figure 5.11, we can see that the RTT observed are different for the various congestion control cases, because they are run one after the other. Instead, in the *parallel* case of Figure 5.12, the RTT curves are very similar, because the flows with different congestion control are run in parallel. The curves are not exactly the same because each IP packet is routed independently, which results in small fluctuations of latency from one packet to the other.

In Figure 5.11, we can observe that Cubic is the congestion control scheme that yields the highest throughput, on average, while BBR adapts better and faster to channel variations. Newreno achieves the highest pick rates, but also suffers the higher variations of throughput and in the use of buffer space, witnessed by the least stable values in the number of bytes in flight. Copa is by far the least performing scheme, as visible, e.g., in the low values reached by its congestion window.

In Figure 5.12, with parallel tests, we can observe that BBR is more stable and adapts quicker to network conditions than the other schemes. This is in contrast with the operation of classic schemes like Cubic an Newreno, which hardly come close to the maximum congestion window size after a change in network conditions. This eventually results also in a better average throughput for BBR, although Cubic obtains comparable results.

With better and more stable network conditions, like shown in Figure 5.13 for a static parallel experiment with a client in Norway, BBR clearly shows its ability to jump fast

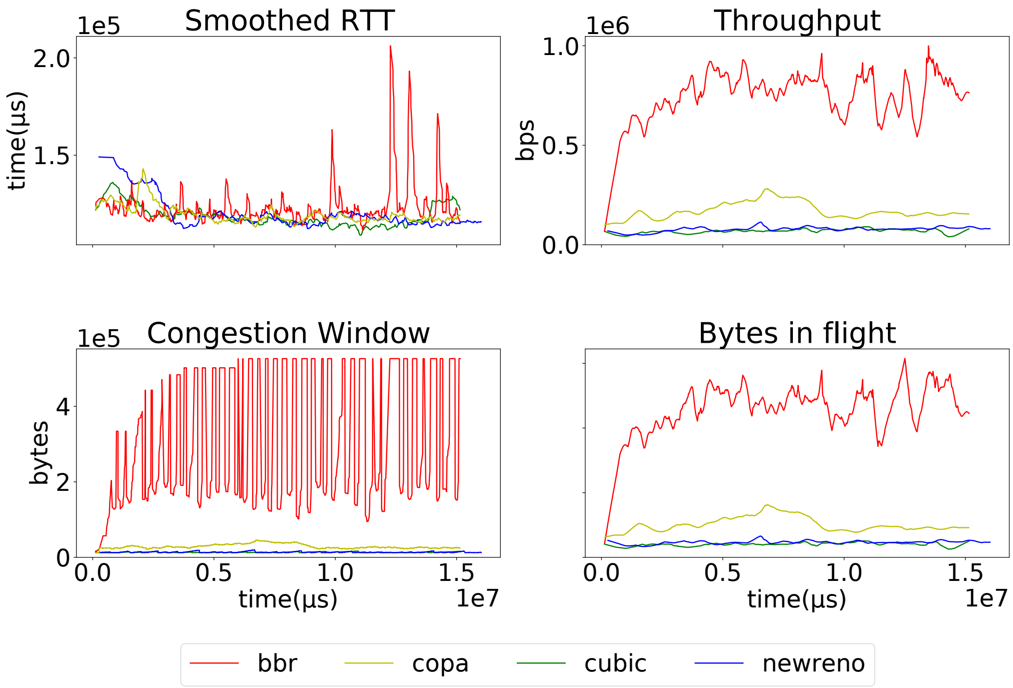


Figure 5.14: Sequential time series in Sweden (with a mobile `Mvfst` client) with three download streams

to better operational conditions with respect to the other schemes. However, it is also interesting to note that Cubic, BBR and Newreno are essentially fair with each other, while Copa exhibits strong limitations.

The fact that BBR is the scheme that succeeds the most at maintaining a sustained throughput value across executions is also visible from experiments with mobile nodes. For instance, Figs. 5.14 and 5.15 show that BBR can introduce oscillations in the congestion window which eventually result in stable and high throughput and number of bytes in flight. This is true both in case of sequential and parallel experiments, which means that the result is not a byproduct of the coexistence of multiple download flows.

Regarding Copa, we remark that we have not optimized its parameters, because that was out of the scope of our measurement study. We leave for future work the investigation on the impact of Copa's configuration on QUIC performance.

5.4. Discussion

We have evaluated the performance of QUIC and HTTP/3 under different cellular network conditions and with multiple congestion control algorithms. We have leveraged the MONROE platform and two open QUIC implementations, namely `flowsim` and `Mvfst`, and employed Docker containers and *qlog*. Containers and raw measurements have been made available online.

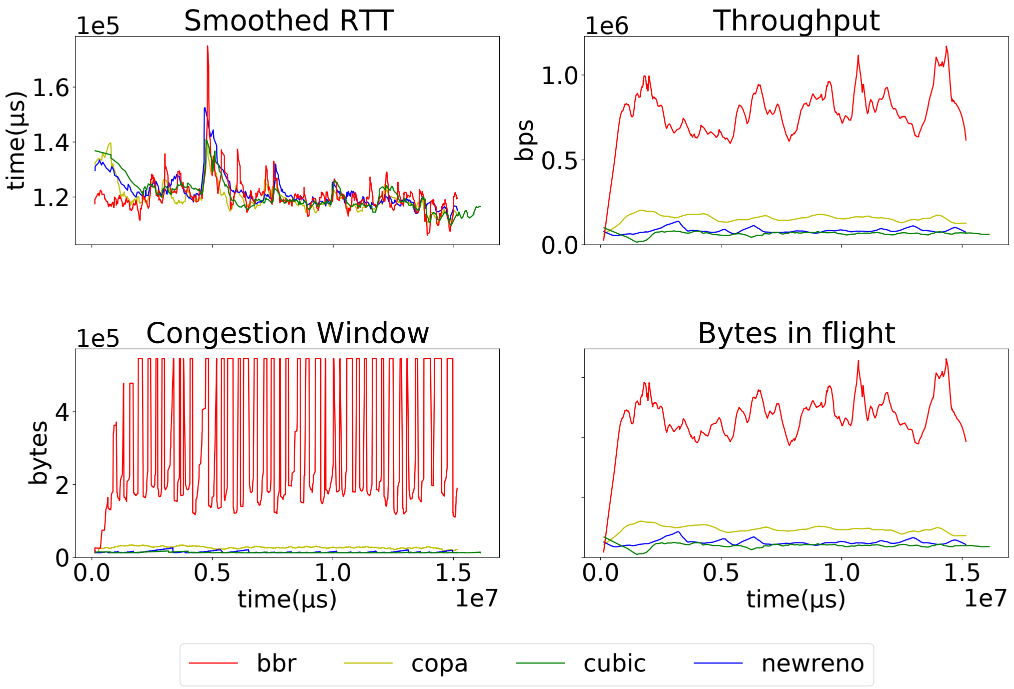


Figure 5.15: Parallel time series in Sweden (with a mobile `Mvfst` client) with three download streams

By analyzing container logs and *qlog* files, we have observed that QUIC is advantageous over TCP for what concerns HTTP applications. In addition, the congestion control algorithm, especially under mobility, strongly impacts the QUIC's overall performance, and BBR yields the more stable performance figures to mobile users.

Our tools and methodology can be used in the future for a broader and continuous assessment of ever-evolving QUIC-based protocols.

Part III

Data science in cellular networks

Monitoring the performance of cellular networks is one part of the story, but one has to wonder what we can do with all of this collected data. Thus, employing data science techniques is essential to obtain meaningful insights from the data acquired. Applying data science has gained momentum in the research community in the past few years due from multiple domains due to the more accessible computational power and cheaper hardware in general. In our case, we use data science to troubleshoot cellular networks; more specifically, we apply interpretable ML techniques, e.g., decision trees, k-means, etc. The fundamental motivation is that troubleshooting cellular networks is still a manual task, assigned to experts who monitor the network around the clock. Besides, existing approaches centered around Explainable Artificial Intelligence (XAI) and based on deep learning and neural networks help to identify problems, but not to interpret and identify the causes of the problems. Indeed, recent studies show that junior engineers can hardly detect network problem with existing tools, and it can take up to 60 minutes for a senior engineer to troubleshoot simple problems. For this reason, in this part of the Thesis, we leverage data collected from real operational cellular networks to present different methodologies to automate the fault identification process in a cellular network and to classify network anomalies based on their inferred causes, using interpretable ML algorithms. In Chapter 6, we present the first stage of the methodology which we call Supervised Trees (STrees). We leverage multiple ML algorithms, mainly Decision Trees and k-means. We test our methodology on several data-set collected by Nokia, and to further validate our methodology, we employ MONROE data-sets. Understanding and classifying anomalous scenarios allows alerting the appropriate department to take corrective actions. Specifically, we use decision trees since they are transparent to inspection, hence interpretable.

In Chapter 6, we rely on a supervised approach, while we later turn our attention to a fully automated, hyper-parameter-free approach. Following this statement, Chapter 7 is a continuation of improving the proposed methodology described in Chapter 6. We introduce a novel feature selection algorithm based on *miscoding*, which is a new metric introduced in this Thesis, and which is based on the concept of Kolmogorov complexity. Each step of the proposed unsupervised methodology uses a specific interpretable algorithm with no hyperparameter. We showcase that STrees and CIAN are easy to deploy in actual production environments, thus accelerating fault detection and troubleshooting cellular networks.

6 Supervised Trees

There has been a remarkable evolution in cellular networks during the recent years. With 4G networks, and even more with the recently roll out of 5G, network services have gained a large degree of intelligence, and involve intensive access to both data communication and computing resources. With the evolution of cellular networks, it has also come an increase in structural complexity and heterogeneity of services, which requires the constant monitoring of the communication system. Indeed, the early detection and correction of operational issues and malfunctioning components in the network is needed to provide network customers with flawless quality of service (QoS) [18]. However, the development and deployment of monitoring subsystems have to face the fast increase in technical complexity of networks [89], and a steady increase in the number and capabilities of mobile devices, hence in the number and complexity of service instances requested to the network [24]. To deal with these phenomena, operators are investing resources into the automation of the maintenance and troubleshooting tasks through self-healing functionalities within the scope of intelligent self-organizing network operation tools. Self-healing network mechanisms are accountable for detecting, identifying, and making decisions on recovery actions [18].

There exist various proposals for making fault detection and self-healing systems effective in mobile networks [19]. However, while traditional approaches lack flexibility and do not scale, newly-defined approaches based on ML lack *interpretability* of results, which hinders the triggering of proper and effective troubleshooting actions when a system fault is detected.

In this work, we join the ML research stream while focusing on the automated detection and classification of possible network performance anomalies. For training and model evaluation, we use real operational network data collected for cellular service auditing purposes by Nokia in various European countries, and we complement them with real operational data gathered by means of the MONROE platform, operating in several European countries as well [90]. Differently from existing proposals, we develop STrees, a supervised ML methodology around an interpretable, cost efficient, scalable, and accessible combination of supervised and unsupervised ML algorithms.

By means of studying the behavior of real networks with respect to TCP performance, the main contribution of this Chapter is a comprehensive supervised ML-based complex methodology that (*i*) identifies if a network is behaving as expected or is under-performing, and (*ii*) automatically determines with high accuracy the root causes that lead to performance issues. In addition, we provide an open source implementation of our methodology, which is based on the use of the Scikit-learn library for Python [91]. Besides, we use valuable real data from commercial networks to test our proposal. The data-set used are webpages such as Facebook, Twitter, Google, and Youtube downloaded while containing TCP statistics, e.g. (RTT, TCP Window, Congestion Window) and radio statistics, e.g. (RSRP, RSRQ, RSSI).

The structure of the rest of the Chapter is as follows. Section 6.1 takes a detailed look into STrees, the supervised ML methodology we propose. Section 6.2 describes our measurements and the data-sets we use. Section 6.3 illustrates the implementation of STrees methodology and analyzes the results it achieves when using real data. We summarize and conclude the Chapter in Section 6.4.

6.1. Supervised ML Methodology

In this section, we present the STrees supervised ML methodology, which we propose to detect and classify networking anomalies at radio access and transport layer. Our methodology is generic for numerical data-sets with multiple features, although here we exploit the Throughput Data Rate (TDR), i.e., the TCP goodput, as a well-defined KPI to characterize anomalies in a data-set reporting TDR and several radio and transport-level observations ("data attributes"). We also leverage the fact that the TDR is strongly correlated to the Round Trip Time (RTT) [92] to build a prediction model.

Figure 6.1 shows the four steps of STrees, starting with (*i*) processing the target KPI to classify available data points according to KPI ranges (this is our "ground truth"). Then we (*ii*) build a classifier for the data points solely based on RTT data attributes to predict TDR values (this is therefore a model), so as to be able to compare the output of the classifier with the ground truth, which leads to detecting anomalies. Thereafter we (*iii*) cluster detected anomalies based on selected attributes (related to either radio or transport layers), which allows to finally (*iv*) classify anomalies based on their cause(s).

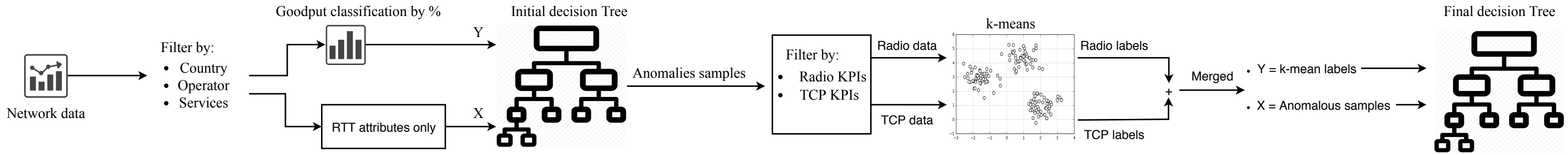


Figure 6.1: High-level presentation of the STrees methodology, starting with the characterization of the target variable, followed by the initial classification of samples using the RTT as the data attribute of choice, grouping anomalies with k-means (with Radio and TCP related data attributes), and final classification of anomalies with a decision tree.

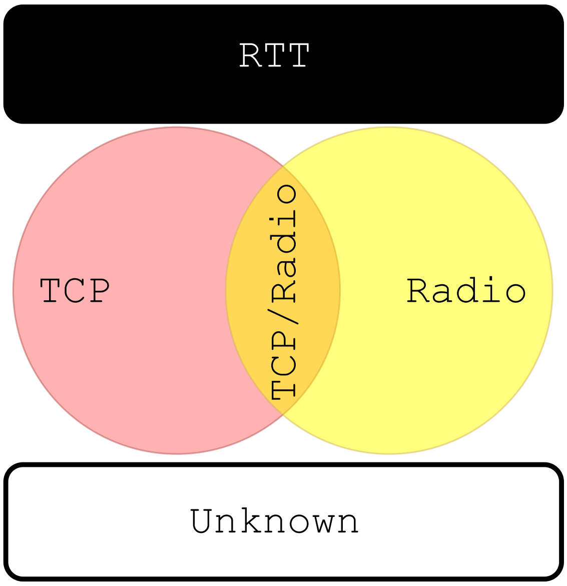


Figure 6.2: Possible causes of a performance anomaly, starting with the RTT as the first choice and moving downwards to identify other factors that cause anomalies in the TDR observed (when using TCP for downloading files). The conclusion may be that the cause of the observed anomalies is not identifiable given the current model.

6.1.1. Target Variable Characterization

The first step of the STrees methodology (see Figure 6.1) starts by characterizing the target variable, the TDR, using a statistical percentiles approach which represents our ground truth on the quality of the TDR. In practice, the data-set is split into three groups: data points with TDR values above the 90-th percentile (Good TDR samples), below the 10-th percentile (Bad TDR samples), and everything else (OK samples). Regulators and self quality assessment teams usually adopt this approach in the analysis of a complex system; in our case, the Nokia team involved in the measurements recommended this approach. Of course, other possible percentile tresholds are possible; for instance, the 20-th and 80-th percentiles could be used. This choice does not affect the proposed methodology.

6.1.2. Detecting Anomalies

After dividing our target variable KPI (TDR), we turn the attention to an interpretable classifier that can classify the TDR into three classes (i.e., Bad, OK, or

Good). There exists a wide range of classifiers fit for this job, but in this particular case, we are looking at an interpretable classifier; for instance, decision trees are humanly interpretable (white box). Thus, we use all the input data to build a decision tree with CART, which classifies the data items into these classes (Bad, OK, and Good TDR), using only the RTT data attributes (e.g., the average/maximum/minimum/standard-deviation RTT observed). We do so since our data-sets contain data about webpage visits, for which it is known that the TDR almost directly originates from the observed RTT [92]. The depth of the tree is limited to $\lfloor (\log_2 n)/2 \rfloor$, where n is the number of samples. Thus rule allows to easily and automatically accommodate different data-set sizes into our methodology.

As a result of the above process, we have a tree that classifies correctly a large portion of the data items. Intuitively, these are experiments in which the values of TDR and RTT are consistent. However, some items are not correctly classified by the tree. We consider these as *anomalies*, since data attribute (RTT) does not explain the target (TDR). For instance, in the next section, we will observe file download and Web page experiments in which TDR and RTT values are inconsistent. We want to explore these data items further, since they may be symptoms of an underlying network problem.

6.1.3. Clustering Anomalies

In the third step of our methodology, we restrict our attention to the anomalous data items misclassified by the initial decision tree. The general idea is to apply a generic clustering technique (i.e., k-means) leveraging Radio data attributes (e.g., Received Signal Strength Indicator (RSSI), Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ)) and TCP data attributes (e.g., congestion window size, receive window size, packet lost values, TCP idle-time) separately, and observe if these attribute combinations yield any groups. An expert supervised the process and handpicked these attributes since the idea is that what cannot be explained within the relation between the TDR and RTT is an anomaly that we should investigate through other data attributes.

In general, we have to identify a number c of potential causes for the misclassification, and for each cause, a set of data attributes that can be used to characterize the problem. As mentioned we use $c = 2$ here and, in the next section, we identified potential anomalies caused by *TCP problems* and *radio problems*. Hence, using only the data attributes that correspond to each potential cause, clustering is used to divide the anomalous data items into two clusters per cause.

By applying this process for each of the c potential causes of problems, we classify anomalous data items into 2^c classes.

6.1.4. Classifying Anomalies

The final step involves building a second decision tree with the full collection of data attributes (e.g., Radio and TCP) identified and trained using the items clustered with k-means in the previous step. The class into which this second tree classifies a data item reflects what makes it to be anomalous, and it is one of the possible 2^c classes identified. We use Gini as the metric to determine the best Radio and TCP data attributes to split the new classes (Radio OK/TCP OK, Radio OK/TCP Problem, Radio Problem/TCP OK, Radio Problem/TCP Problem) at each point. For instance, in the example presented in the next section, the tree obtained in the final step determines if a given TCP flow is anomalous because of TCP problems or radio problems (or both, or none, see Figure 6.2). Observe that the outcome may show that there are several causes for the same data item. It is also possible that no cause is assigned to a data item, because it can be a false positive or because the actual cause is not among the set of c considered causes (TCP or Radio, in our case).

In STrees, we leverage the interpretability of decision trees to find the conditions in the attributes that make each data item anomalous. The path from the root to the leaf gives these conditions in the decision trees. This is expected to be useful for network administrators to identify what is causing the anomaly and allow for a fast diagnosis and solution of the corresponding network problem. Observe that this decision tree can also be used in the future to classify other anomalous data items. Suppose the network administrators have been able to identify the issues that made an experiment anomalous in the past. In that case, they can use that experience with new instances, and very possibly fix the problem quickly.

6.2. data-sets

In this section, we present the data-sets used in this study, namely from MONROE and Nokia.

6.2.1. Nokia drive-test measurements

To benchmark quality of service in mobile networks, continuous drive tests with end-to-end test scenarios are performed every day internally by the network operators (Quality Teams), and externally by third-parties or government regulators. Each of these test campaigns can have up to tens of thousands of individual test cases, from which specific metrics are calculated. These drive tests are normally executed with off-the-shelf testing equipment (NEMO, TEMS, Swissqual, etc.), capable of running predefined sequences of tests and collecting relevant low level radio and traffic information, as well as application performance statistics. Part of the data-sets used for this experimental validation are real test records used by Nokia for the assessment of various mobile networks in year 2019. The data used in this study has been generated by processing all the information provided by the testing equipment and aggregating it at test level (count, sum, min, max, average, percentiles, etc.). The resulting data-sets have a single row per test and hundreds of columns summarizing all the dimensions (date, time, location, network element information, etc.) and features (radio, TCP/IP, application, etc.) related to that particular test. The data transmitted in the drive tests is synthetic and does not include any customer's sensitive information, hence respecting the European Union General Data Protection Regulation (GDPR) and similar regulations (i.e., the data has been generated by a testing device and not by real users). Finally, potentially commercial sensitive data, such as the identity of the network operator, has been anonymized.

The Nokia-provided data-sets consists in 2.2 GB of data in CSV format, containing 358 514 rows and 1164 attributes per row, obtained from direct measurements and post

Table 6.1: Overview of MONROE data-sets collected in Norway, Sweden, Italy, and Spain

# Services	Countries	# Measurement nodes	# Months	Size (GB)
Facebook	all	239	24	11.0
Google	all	240	24	10.0
Youtube	all	239	19	5.1
Twitter	all	231	18	3.1

processing of pcap files, which produced several conditional statistics on the selected performance indicators. The pcap files themselves are not included in the database. For instance, the drive-test data-set reports separate statistics for the throughput and RTT observed over the entire download of a file or over the initial n seconds, for multiple values of n. This data-set is also rich in terms of metadata, which allows us to filter experiments by test type, infrastructure, operator, vendor, device type, and communication technology, among other parameters.

6.2.2. MONROE measurements

In addition to the data provided by Nokia, this study also leverages MONROE data-sets. Table 6.1 reports a breakdown of the data collected with MONROE for the services used for this study for all operators[1] available in their respective countries. We divide experiments into mobile vs wired connections. Overall, We have three mobile network operators and a wired connection for each node, although we only use one operator at a time to avoid interference and overloading the node's board. For the experiments analyzed in this Chapter, we consider the historical data collected by several MONROE nodes in four European countries over several months, using active experiments and generating statistics with Tstat. Each node performed different experiments, which caused them to contact Facebook, Twitter, YouTube, and Google. The data collection duration varies from 19 months to 23 months, depending on the country and service, and the node type (static or mobile) used. The database with the services used in this extension for all countries and operators totals 29.2 GB in CSV format, with 144 attributes comprising millions of TCP flows.

[1] We do not report the name of the operator for the sake of privacy.

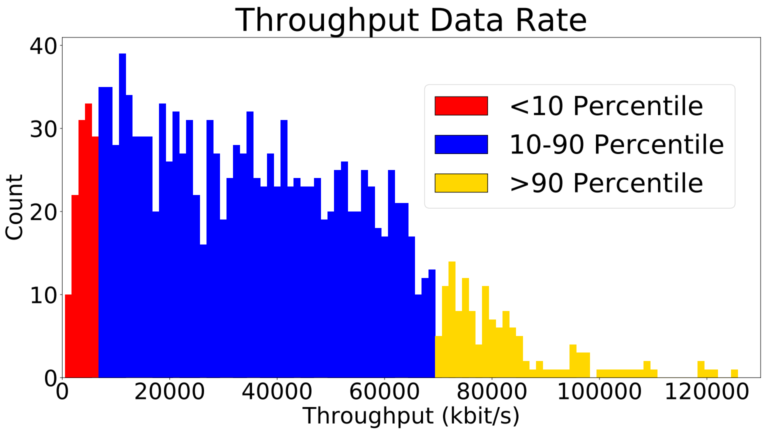


Figure 6.3: Distribution of Throughput Data Rate.

6.3. Results

In this section, we report the results of our methodology, implemented by using the widely adopted scikit-learn library from Python in real data. Scikit-learn provides ML algorithms, such as decision trees and k-means.

6.3.1. Data Analysis for Nokia Drive Tests

We have only chosen one value from each of categories available in the Nokia drive-test data-set. This suffices to showcase how our methodology works without losing focus in the description of experiments, and to demonstrate the properties of our proposal. The data-sets used correspond to Hypertext Transfer Protocol (HTTP) file downloads, in cities, over LTE networks, and with one single operator (we anonymize the operator's name for privacy purposes). Filtering the data with these parameters produced 1494 samples. Henceforth, these are the data items (samples, experiments) to be used and analyzed in this subsection.

6.3.1.1. TDR Characterization

Figure 6.3 depicts the distribution of the TDR values. Using these TDR values, we split the samples into three groups: those with TDR values above the 90-th percentile (Good throughput samples), below the 10-th percentile (Bad samples), and everything

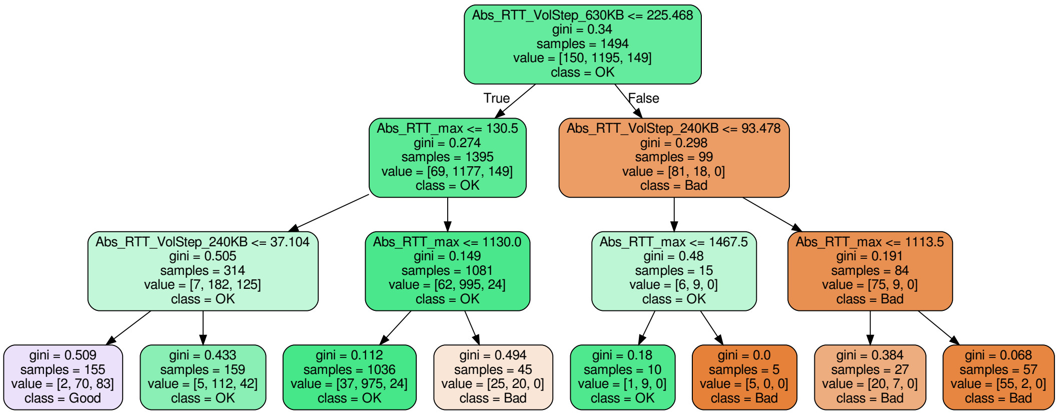


Figure 6.4: Decision tree generated with supervised ML using RTT attributes as input and TDR classes inferred from percentiles (see Figure 6.3). when visualizing the tree each box has the quality of the gini split, the number samples at each split, the number values for each TDR class (Bad, OK, Good), and which class was picked.

else (OK samples). Specifically, the value of the 10-th percentile was 7 796 kbit/s, while the value of the 90-th percentile was 68 493 kbit/s. This approach based on the use of statistical percentiles is commonly adopted by regulators and (self-) quality assessment teams for the analysis of complex systems, and by the Nokia team involved in the measurements.

6.3.1.2. Detecting Anomalies

We use the TDR split of the 1 494 samples as a target for a supervised ML decision tree with three classes: Bad, OK, and Good. Since it is well known that the TCP throughput is a function of the RTT, in the classification we use only the RTT attributes available in the data-set to feed the CART algorithm. To avoid overfitting we limited the number of internal tree levels to three. Figure 6.4 illustrates the resulting decision tree with RTT attributes. For a given sample, at the root of the node, it is decided if the Abs_RTT_VolStep_630KB attribute (the RTT after downloading 630 KB) value of the sample is smaller or equal to 255.468 ms. If true, we move to the left side of the tree, and to the right side otherwise. Other information included in each tree node besides the attribute and split value is the actual value of the Gini impurity for the class split at

Table 6.2: Summary of the decision tree rules of Figure 6.4 producing the most significant RTT attributes against the TDR percentile class split shown in Figure 6.3

Rules	**Prob.**	**Class**
If Abs_RTT_VolStep_630KB <= 225.468	0.84	TDR OK
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max <= 130.5	0.58	TDR OK
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max <= 130.5 and if Abs_RTT_VolStep_240KB <= 37.104	0.54	TDR Good
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max <= 130.5 and if Abs_RTT_VolStep_240KB >37.104	0.70	TDR OK
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max >130.5	0.92	TDR OK
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max >130.5 and If Abs_RTT_max <= 1130.0	0.94	TDR OK
If Abs_RTT_VolStep_630KB <= 225.468 and If Abs_RTT_max >130.5 and If Abs_RTT_max >1130.0	0.56	TDR Bad
If Abs_RTT_VolStep_630KB >225.468	0.82	TDR Bad
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB <= 93.478	0.60	TDR OK
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB <= 93.478 and If Abs_RTT_max <= 1467.5	0.9	TDR OK
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB <= 93.478 and If Abs_RTT_max >1467.5	1.00	TDR Bad
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB >93.478	0.89	TDR Bad
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB >93.478 and If Abs_RTT_max <= 1113.5	0.74	TDR Bad
If Abs_RTT_VolStep_630KB >225.468 and If Abs_RTT_VolStep_240KB >93.478 and If Abs_RTT_max >1113.5	0.96	TDR Bad

this level of the tree, the total number of samples considered in the node (1 494 at the root) and the number of samples for each label [150, 1195, 149], that correspond to the classes [Bad, OK, Good]. The class value matches the predicted class at this level of the tree (the class that has a highest number of samples). Employing a tree traversal, we will reach a leaf node, where there are no more conditions, and the class for the sample

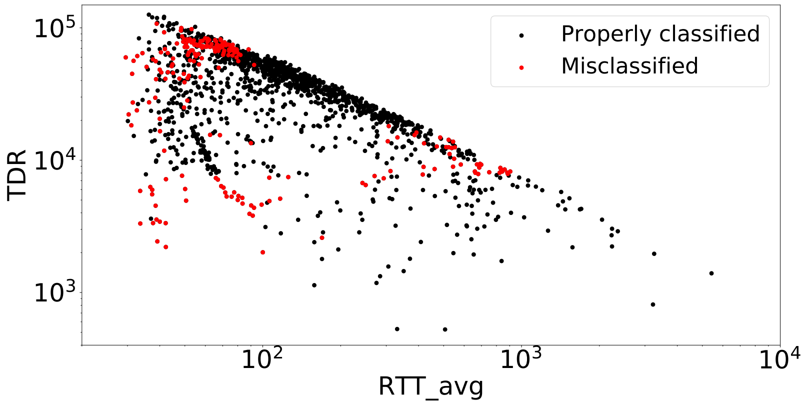


Figure 6.5: Graphical plot of the data items properly classified by the decision tree of Figure 6.4 in black, and the misclassified data items in red.

Table 6.3: Confusion matrix for the supervised ML classifier of Figure 6.4 trained with the classes identified using Figure 6.3

	Bad (ML)	**OK (ML)**	**Good (ML)**
Bad (percentile)	105	43	2
OK (percentile)	29	1096	70
Good (percentile)	0	66	83

is defined. The decision tree depicted in Figure 6.4 was able to classify the TDR samples to some degree, with an accuracy score of 85% (number of correct predictions from the TDR classes computed by the tree, in this case 1 269, over the total number of samples, which is 1 494). Table 6.2 describes the results generated by the decision tree at relevant nodes.

After using the decision tree built with the CART algorithm for the classification of all the samples in the data-set, we still have 225 samples, which represent 15% of the original data-set, that are not correctly classified (they are visually reported in Figure 6.5). More in detail, Table 6.3 reports the confusion matrix, which shows how data samples labeled by means of percentile tresholds are subsequently classified by the decision tree, based on RTT attributes only. Understanding the cause of these misclassification is the target of the methodology presented in the previous section. We therefore consider next how to cluster these anomalous samples (i.e., the data samples that are incorrectly modeled and classified with the decision tree).

Table 6.4: radio and TCP data attributes used by the unsupervised ML algorithm used in STrees (i.e., k-means, with $k = 2$)

Type	**Attributes**	**Description**
Radio	Start.RSSI.dBm	Received signal strength indication initial value.
Radio	End.RSSI.dBm	Received signal strength indication (final value).
Radio	Start.RSRP.dBm	Reference Signals Received Power (initial value).
Radio	End.RSRP.dBm	Reference Signals Received Power (final value).
Radio	Start.SINR.dB	Signal-to-Interference-plus-Noise Ratio (initial value).
Radio	End.SINR.dB	Signal-to-Interference-plus-Noise Ratio (final value).
TCP	Abs_CWIN_avg	Average congestion window size.
TCP	Abs_CWIN_max	Maximum congestion Window size.
TCP	Abs_RWIN_avg	Average TCP receive window.
TCP	Abs_RWIN_max	Maximum TCP receive window.
TCP	Abs_PacketLost_sum	PacketLost total value.
TCP	Abs_IdleTime_avg	Average gap value between consecutive TCP segments.
TCP	triple_dupacks_b2a	Triple duplicate ack value (server to client).

6.3.1.3. Clustering Anomalies

To cluster the misclassified samples obtained in the previous step we have chosen k-means to use as technique, due to its suitability for a medium-sized data-set and the ability to cluster data. We fit k-means with attributes picked from a homogeneous class of data attributes. In particular, we use either TCP-related or radio-related attributes. We try to identify if the problem belongs to TCP events (losses, duplicated ACks, etc.,), radio quality events (e.g., changes in signal strength), or a combination of both. Table 6.4 displays the data attributes used.

Although not presented here, we have tested various possibilities for the numbers of target clusters, and found out that the highest score of k-means was obtained by using only two clusters. Indeed, the use of k-means revealed that the data incorrectly classified by the decision tree can be further clustered into two groups according to TCP attributes. Similarly, the best choice is to use two clusters also in case of using radio attributes.

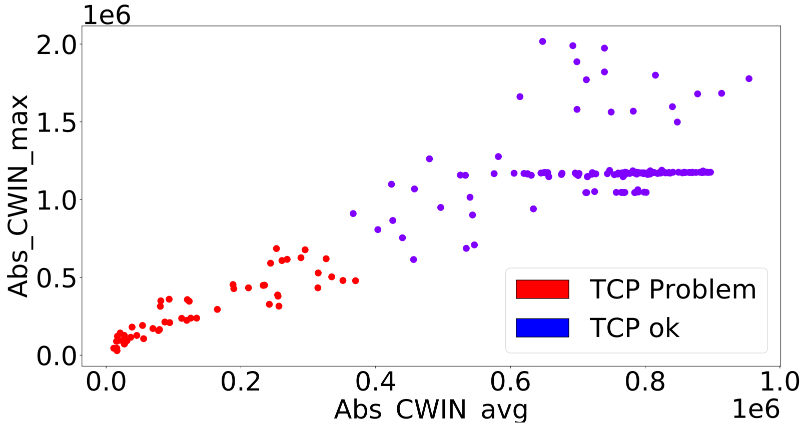


Figure 6.6: The two k-means clusters obtained with TCP data attributes with respect to the congestion window average and maximum value attributes.

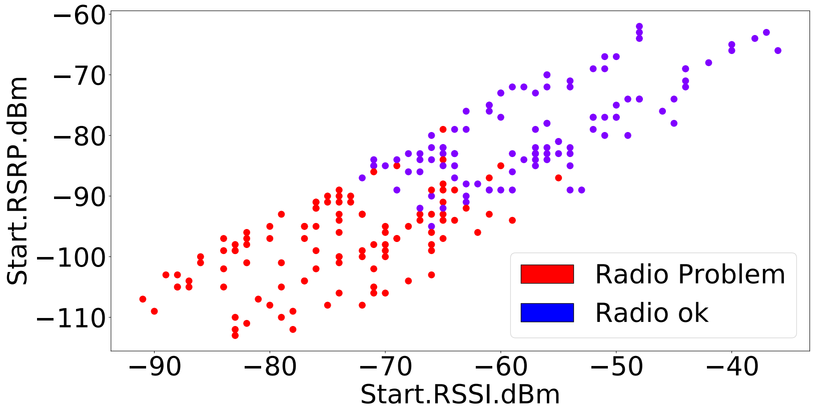


Figure 6.7: The two k-means clusters obtained with radio attributes with respect to the RSSI and RSRP attributes.

Figure 6.6 and Figure 6.7 show the clusters obtained by using carefully chosen pairs of attributes. The first figure reports an example of TCP attributes and the second depicts an example of radio attributes. In both cases, it is clear that the clusters obtained with k-means separate the samples into those that have TCP (resp., radio) issues and those that do not have issues. Therefore, applying k-means to TCP (resp., radio) attributes allows to identify whether there exists a problem with the TCP (resp., radio) performance.

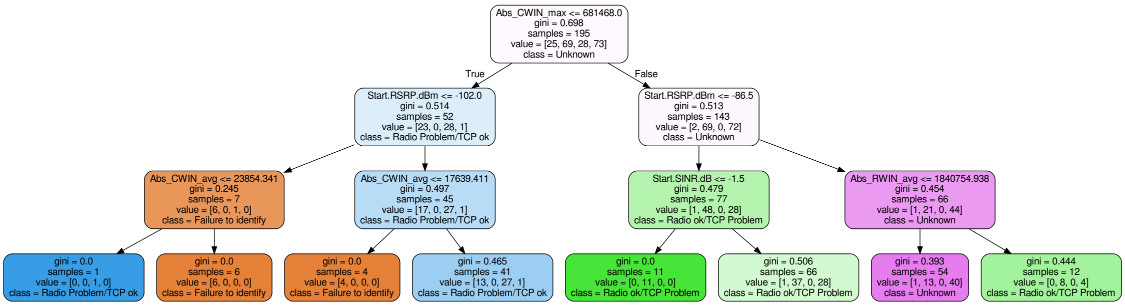


Figure 6.8: STrees anomaly detection decision tree with TCP, radio attributes from Table 6.4, and k-means labeled clusters as classes. when visualizing the tree each box has the quality of the gini split, the number samples at each split, the number values for each class (Failure to identify, Radio Ok/TCP Problem, Radio Problem/TCP Ok, unknown), and which class was picked during each split.

6.3.1.4. Classifying Anomalies

The last stage of our methodology involves deriving a final model, using again a decision tree, to identify the root cause of the identified anomalies. For our specific case study, we use a decision tree to classify into the cases that the causes of the anomaly are TCP events, radio conditions, or both. This decision tree has been trained with the k-means labels previously collected from TCP-based and radio-based clustering, and a new collection of relevant attributes (see Table 6.4). Figure 6.8 showcases the resulting decision tree, which not only classifies TCP and radio issues, but also identifies a class of anomalies that cannot be explained by means of TCP and radio attributes (labeled as “failure to identify”). A problem is classified as “unknow” when it is not possible to distinguish between TCP or radio problem, but it is definitely one of these two. An enumeration of the rules applied by the decision tree can be found in Table 6.5. Figure 6.9 depicts the points that belong to each class in terms of TDR and RTT. Observing the final decision tree, we can distinguish if the problem is due to TCP with a probability of 0.623, or radio with probability of 0.60 (note that events are not mutually exclusive so that their probabilities do not need to sum up to one or less). The overall score of the tree is 70%, which means that 70% of the anomalies are identified, jointly with their root causes. Misclassification is mostly due to lack of data, since after three split levels, the amount of data at some leaf nodes is no longer statistically relevant.

Table 6.5: Highlights of the STrees decision tree rules for detection of anomalies in Figure 6.8 showcasing the dominant attributes from Table 6.4 and classes from Figure 6.8

Rules	**Probability**	**Class**
If Abs_CWIN_max <= 681468.0	0.54	Radio Problem
If Abs_CWIN_max <= 681468.0 and if Start.RSRP.dBm <= -102.0	x	Failure to identify (lack of data)
If Abs_CWIN_max <= 681468.0 and if Start.RSRP.dBm > -102.0	0.60	Radio Problem
If Abs_CWIN_max <= 681468.0 and if Start.RSRP.dBm > -102.0 and If Abs_CWIN_avg >17639.411	0.66	Radio Problem
If Abs_CWIN_max >681468.0	x	Unknown problem! (investigate further)
If Abs_CWIN_max >681468.0 and if Start.RSRP.dBm <= -86.5	0.62	TCP Problem
If Abs_CWIN_max >681468.0 and if Start.RSRP.dBm > -86.5	0.67	Unknow problem! (investigate further)

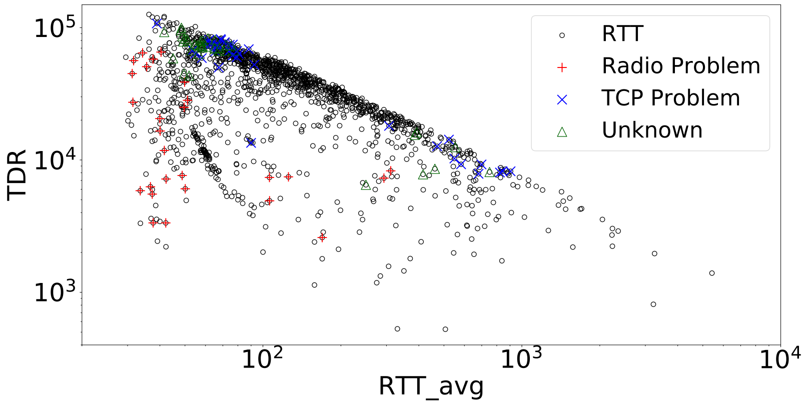


Figure 6.9: The outcome of the misclassified points using a combination of unsupervised and supervised ML in Figure 6.8 and properly classified points with the supervised ML classifier in Figure 6.4.

Note that the “Failure to Identify” class shown in the decision tree of Figure 6.8 and in Table 6.5 is due to a lack of training data over other attributes. For example, the availability of Domain Name System (DNS) and Transport Layer Security (TLS) measurements might help to identify more types of network problems and further complement the analysis in a fully automated way.

6.3.2. Data Analysis for MONROE data-sets

The MONROE data-sets come in as a layer to extend and validate our methodology. We choose multiple data-sets with multiple services such as Facebook and Google collected in various European countries ranging from south (Spain, Italy) to north (Sweden, Norway) using a range of mobile operators collected from an experiment campaign in the MONROE platform. The methodology is the same with minor tweaks; for instance, we change the TDR characterization from 90-10 to 80-20 percentiles to validate the claim that choosing other percentile tresholds does not affect the proposed methodology. Thus, everything above the 80-th percentile is considered Good TDR, below the 20-th Bad, and in between (20-th to 80-th) OK TDR. We test the methodology on three operators per country for a given service. We report the results and interesting findings per service.

6.3.2.1. Facebook

The first test in question is with Facebook. We conducted multiple Facebook webpage downloads in various European countries. The average downloaded webpage size of Facebook was 72 757.3 bytes and, at each experiment run, we cleaned the cache to make sure that a new complete download occurs. In some countries, such as Sweden and Norway, we had access to mobile nodes mounted on public transport vehicles. Thus, the downloaded page statistics for those countries are a mixture of statistics for static and mobile cellular nodes, hence the slightly higher radio problems which is depicted later in this subsection. As for Italy and Spain, we only report results for static cellular nodes.

The data-set consisted of 176 018 samples. As an example, Figure 6.10 exhibits the TDR values distribution for the data-set obtained using an anonymised *operator 0* in Sweden. Notice that the 20-th and 80-th percentiles are defined for each studied group of data, and if we consider the entire subset of operators and countries monitored in the study, the percentile treshold values range from 500 kbit/s up to 60 Mbit/s. Our methodological procedure uses these tresholds on the TDR split with a decision tree, while fitting only the Tstat server-side RTT attributes as the input features to classify the TDR.

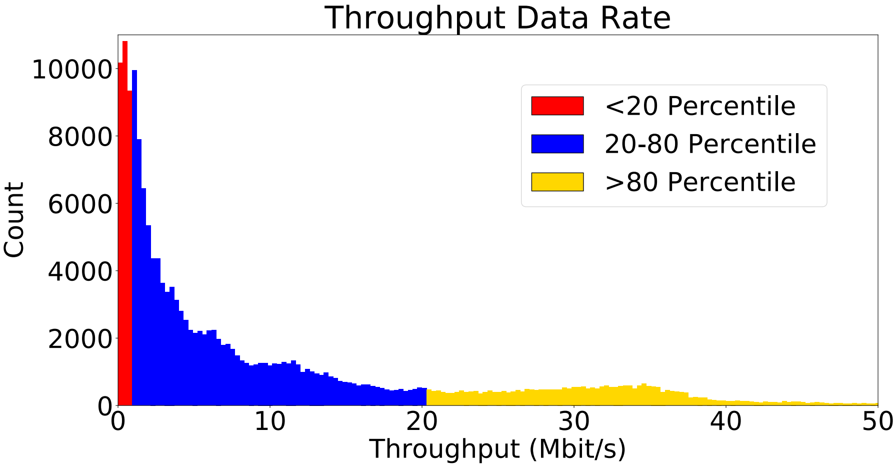


Figure 6.10: A snapshot of the Distribution of Throughput Data Rate in Sweden using Operator 0 and Facebook as a service.

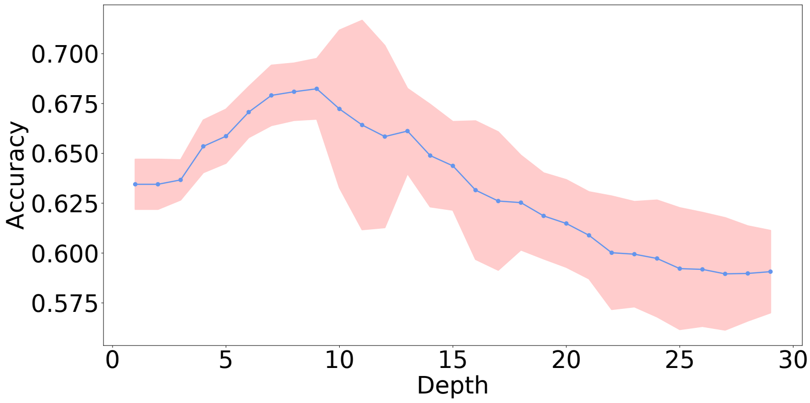


Figure 6.11: The depth versus accuracy for a decision tree with the confidence interval using Facebook as service in Sweden with Operator 0.

The decision tree depth now changes depending on how many samples are available, and for the case of *operator 0* in Sweden, the depth was $n = 6$. As a proof that our rule on the selection of n works, Figure 6.11 depicts the depth versus the accuracy alongside the confidence interval applying cross-validation 30 times, again for *operator 0* in Sweden. From the figure, it is clear that with a depth of 6 we gain more accuracy while we avoid overfitting. We repeat these steps, not only for one operator in Sweden but also for all operators available in the data-set.

Table 6.6 reports results obtained for the first decision tree, accompanied by their accuracy, precision, recall, and F1 score. To get the precision, the recall, and the F1

Table 6.6: A summary of the initial decision tree's performance utilizing Facebook as a service in all the countries part of the MONROE project with all their operators

Service	Country	Operator	Accuracy (%)	Precision (%)	Recall (%)	F1 Score	samples	anomalies
Facebook	**Sweden**	*op*0_sw	68.1	66.0	67.8	64.3	176018	55784
		*op*1_sw	66.0	61.7	64.8	59.0	137777	46814
		*op*2_sw	70.0	67.0	68.6	65.4	132802	40841
	Norway	*op*0_no	73.1	71.9	72.2	71.5	111276	29904
		*op*1_no	74.5	74.4	74.4	73.4	132128	33233
		*op*2_no	73.2	72.1	72.7	71.3	112728	29611
	Italy	*op*0_it	75.0	73.0	73.3	71.0	48265	12117
		*op*1_it	75.6	75.4	75.3	74.7	73352	17869
		*op*2_it	70.6	67.3	67.6	66.4	36223	10622
	Spain	*op*0_es	74.7	73.6	73.4	71.3	27289	7025
		*op*1_es	76.4	75.5	76.2	75.2	75903	17878
		*op*2_es	72.5	70.9	70.7	67.8	23449	6442

Score, we split the data-sets into training and test subsets, with 20% testing data.

Following the methodology described in Section 6.1, we identify anomalous samples and use them to fit k-means with clusters leveraging radio and TCP attributes. For matching, we use Tstat reported attributes on radio and TCP, such as RSSI, RSRP, and RSRQ in the case of radio and Congestion and Receiver Window in TCP. Figure 6.12 takes a closer look at the clusters detected by using radio attributes, where we can see that using k-means produces two sets of groups; TCP follows the same trend in that regard, as shown through Figure 6.13. We further summarize the results with the counts of radio and TCP problem observed in the Facebook experiments with the final step of our methodology in the topmost part of Table 6.7. Depending on the country and their operator, the ratio between radio and TCP problems might vary. For instance, in Sweden and Norway we can observe more radio problems than in other countries, which is due to the fact that MONROE nodes used in those countries were not only static nodes (e.g., in labs and office) but also mobile nodes (e.g., moving with buses and trains).

We can therefore deduce that our methodology can detect anomalies, and their causes, starting from the analysis of very different data-sets. Indeed, our methodology easily adapts to the structure and size of the chosen data-set. We further corroborate our intuition by reporting results on different data-sets and experiment, starting with the interesting finding encountered when testing the Google service, described next.

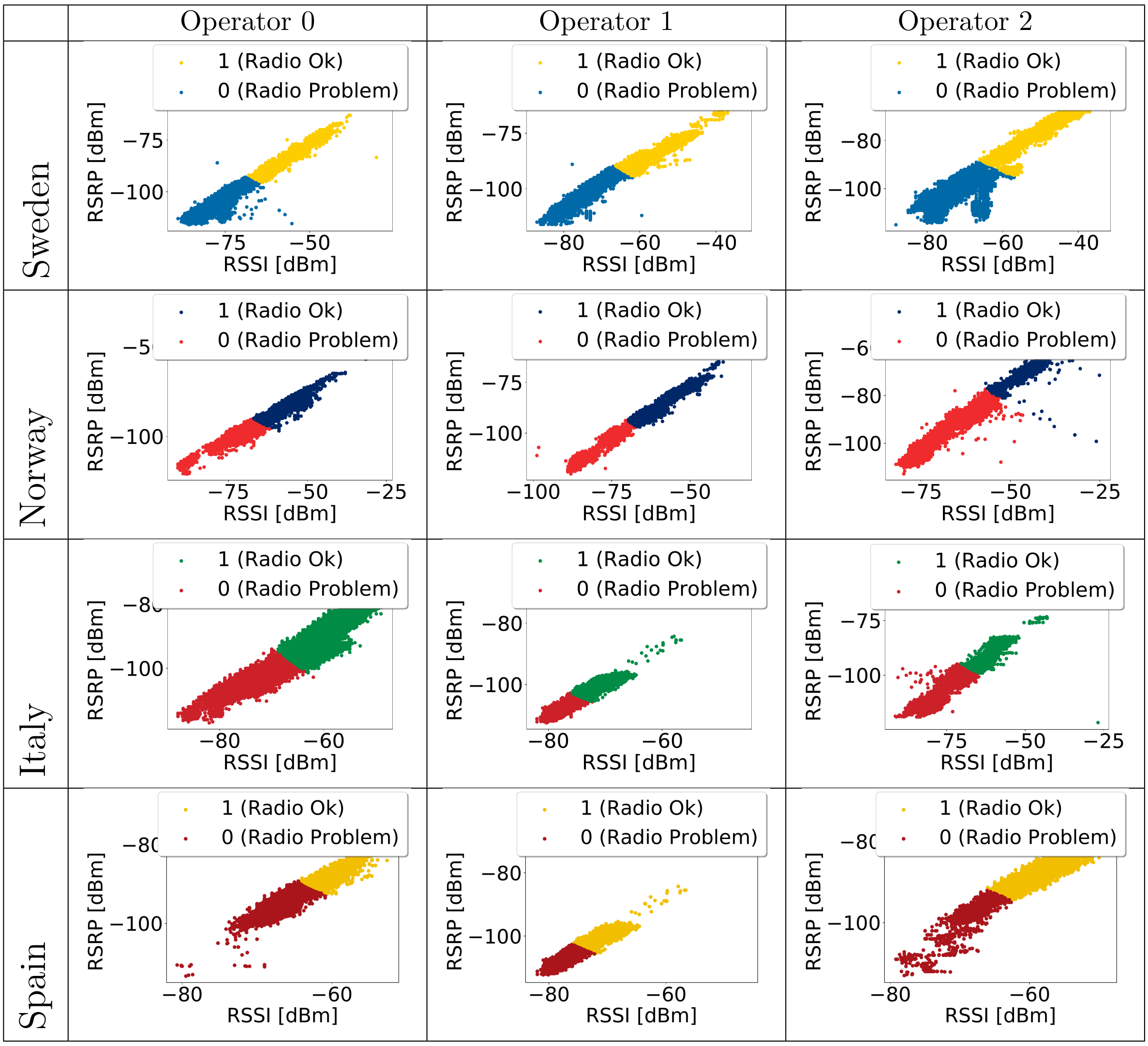


Figure 6.12: The two k-means clusters per countries and operators obtained with radio attributes concerning the RSSI and RSRP attributes using anomalies sample from table 6.6 leveraging Facebook as a service.

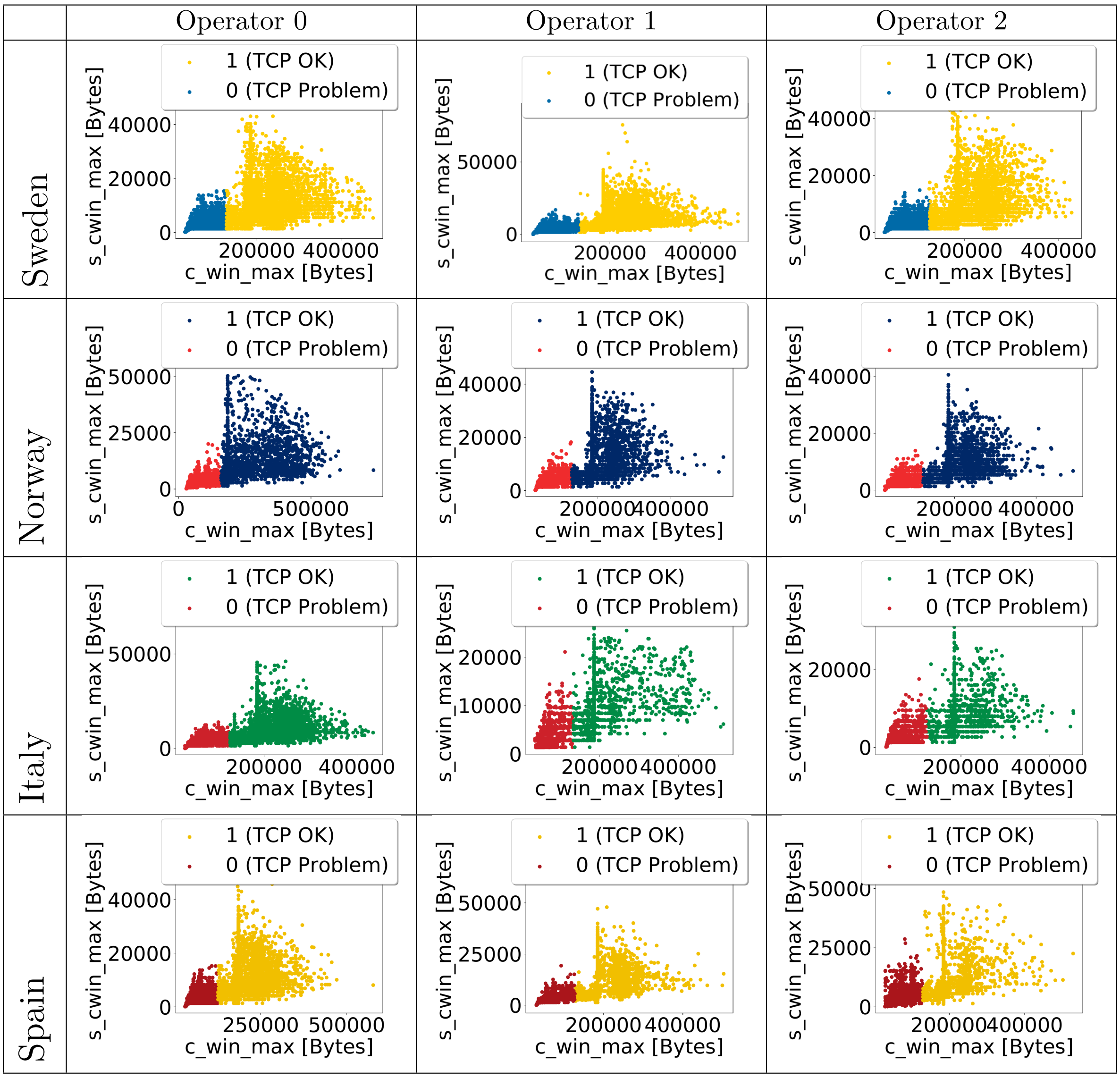


Figure 6.13: The two k-means clusters per countries and operators obtained with TCP attributes using anomalies sample from table 6.6 leveraging Facebook as a service

Table 6.7: The final decision tree performance with STrees, using MONROE data-sets for Facebook and Google experiments.

Services	Country	Operators	Accuracy (%)	Precision (%)	Recall (%)	F1 Score	Radio Problem	TCP Problem	Prob in both
Facebook	**Sweden**	*op*0_sw	99.1	98.0	96.0	98	13082	5153	2963
		*op*1_sw	98.7	97.2	95.2	96.1	5914	8065	5311
		*op*2_sw	99.0	98.2	97.4	98.3	19416	2487	4416
	Norway	*op*0_no	99.3	99.2	98.7	98.9	10747	3188	2990
		*op*1_no	99.3	99.2	98.3	98.0	12136	2384	2877
		*op*2_no	99.0	98.0	97.0	97.0	3304	11924	9797
	Italy	*op*0_it	99.4	98.4	97.2	98.0	1642	6750	4425
		*op*1_it	98.7	92.6	94.4	92.5	801	2880	5194
		*op*2_it	99.5	99.0	99.0	99.0	372	5357	1884
	Spain	*op*0_es	98.3	98.5	97.0	96.0	2631	5039	9008
		*op*1_es	99.3	98.2	98.1	98.0	656	3544	2314
		*op*2_es	99.0	98.3	98.5	97.7	1046	1057	4493
Google	**Sweden**	*op*0_sw	99.8	99.9	99.9	99.9	11120	1	122
		*op*1_sw	99.7	99.7	99.7	99.7	22799	2	203
		*op*2_sw	99.4	99.6	99.6	99.4	9653	3	106
	Norway	*op*0_no	99.3	99.5	99.5	99.5	4852	681	637
		*op*1_no	99.4	99.7	99.7	99.7	7839	8	5
		*op*2_no	98.5	98.7	98.7	98.7	10591	333	422
	Italy	*op*0_it	99.3	99.5	99.5	99.5	12121	69	23
		*op*1_it	99.7	99.7	99.7	99.7	7141	118	71
		*op*2_it	98.4	99.2	99.2	99.2	4591	476	372
	Spain	*op*0_es	98.7	98.5	98.9	98.9	14058	30	716
		*op*1_es	98.7	98.9	99.0	99.0	4227	23	-
		*op*2_es	99.0	99.7	99.7	99.7	1586	8	2

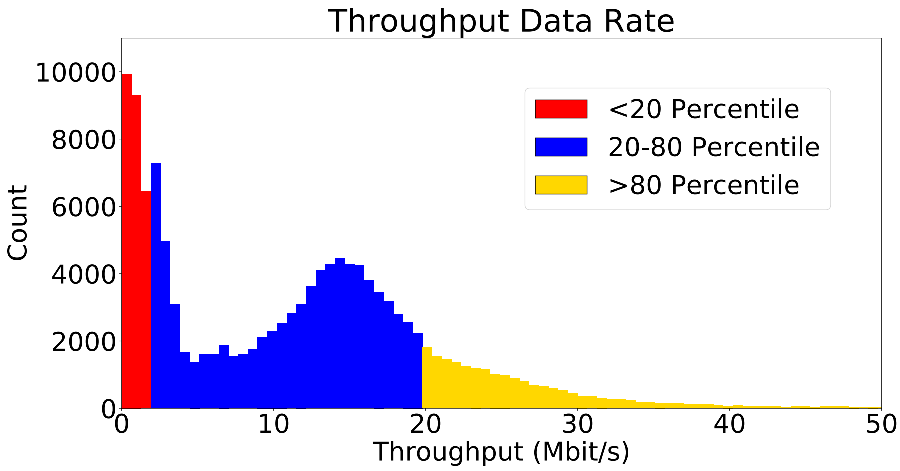


Figure 6.14: A snapshot of the Distribution of Throughput Data Rate in Sweden using Operator 0 and Google as a service.

6.3.2.2. Google

A second experiment with MONROE and Tstat data leverages the Google search engine as the downloaded webpage. We observed an average downloaded size of 17 368.8 bytes, following the same procedure as the Facebook service (repeating downloads, clearing the cache, etc..). The available TDR samples are classified as shown in Figure 6.14, while the first decision tree with the TDR as classes and RTT as input features yields the results reported in Table 6.8.

Table 6.8: A summary of the initial decision tree's performance utilizing Google as a service following up the same methodology steps from section 6.3.2.1

Service	Country	Operator	Accuracy	Precision	Recall	F1	samples	anomalies
			(%)	(%)	(%)	Score		
Google	**Sweden**	*op*0_sw	64.8	66.0	67.7	65.4	100876	35475
		*op*1_sw	66.0	66.0	67.7	65.5	97945	34760
		*op*2_sw	70.0	66.8	68.6	64.6	90198	31550
	Norway	*op*0_no	66.9	70.2	69.9	66.3	52158	17228
		*op*1_no	67.6	71.4	70.3	67.8	42028	13586
		*op*2_no	66.8	69.7	69.9	67.6	54996	18261
	Italy	*op*0_it	67.7	72.1	71.2	68.9	74596	24091
		*op*1_it	66.5	69.0	69.0	66.2	46978	15726
		*op*2_it	66.0	69.3	67.6	66.4	32909	11164
	Spain	*op*0_es	69	71.9	71.7	70.5	75279	23334
		*op*1_es	66.0	70.5	69.7	66.6	31863	10821
		*op*2_es	64.8	68.4	68.3	67.8	27058	8315

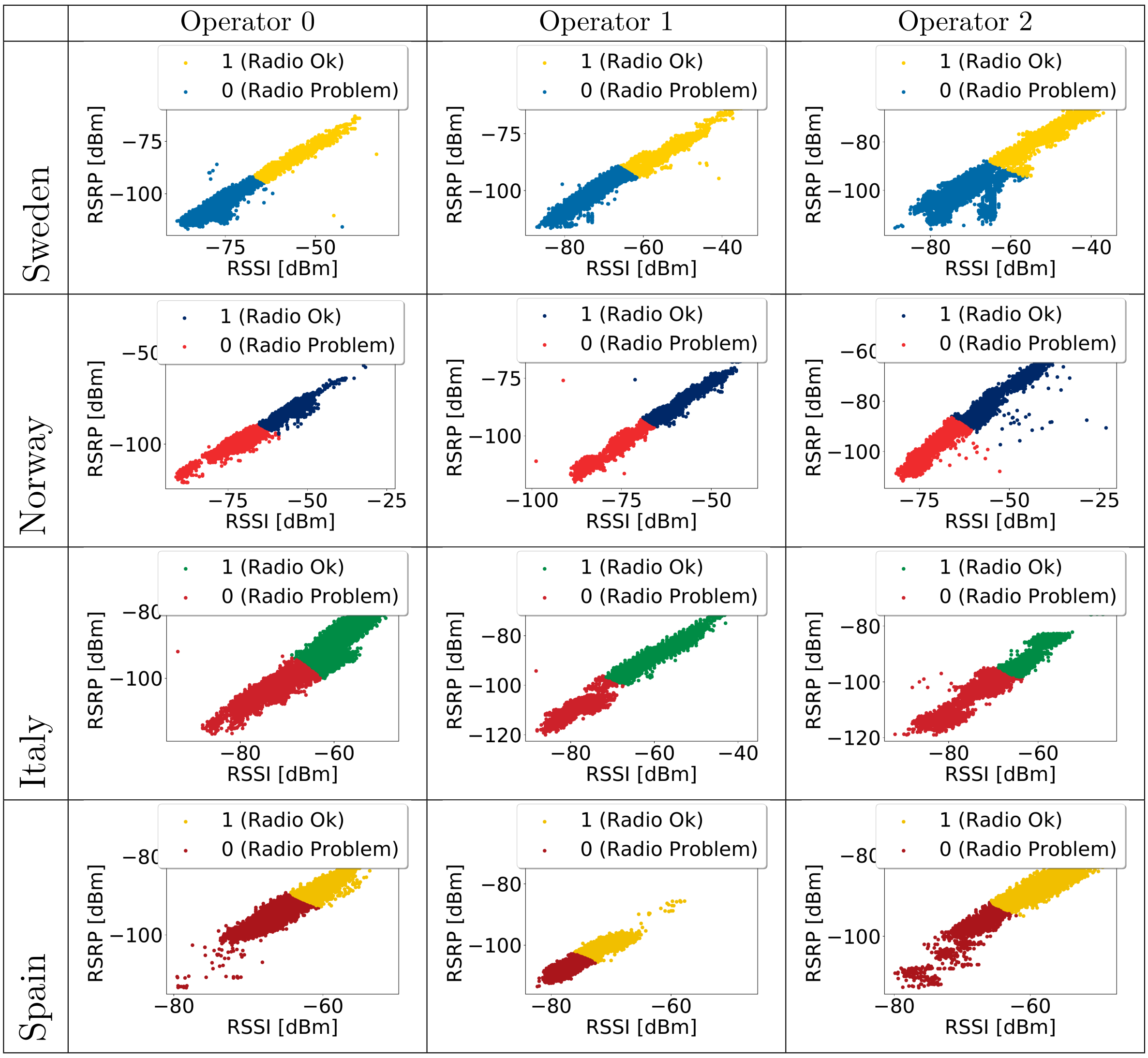


Figure 6.15: The two k-means clusters per countries and operators obtained with radio attributes concerning the RSSI and RSRP attributes using anomalies sample from table 6.8 leveraging google as a service.

The interesting part comes in treating the anomalies samples from Table 6.8 with k-means. In the radio case shown in Figure 6.15, k-means produced two sets of clusters separated, which is consistent with what previously observed in this Chapter with other data-sets. However, a neat change comes in the TCP case where k-means could not identify two clusters. This is logical and somehow expected given (i) the small size of downloaded webpage, and (ii) the fact that Google services are replicated very close to users. Indeed, moving forward to our methodology's final step, we can see from Table 6.7 that the number of TCP problems in the Google experiment is low or non-existent;

instead, we have mainly radio problems, which do not directly depend on the nature of the downloaded page and its location. This indirectly confirms that our methodology finds meaningful conclusions on the origins and causes of anomalies.

6.3.2.3. Other services

We have used our methodology to analyze more data-sets collected with MONROE and Tstat, e.g., by repeatedly accessing the webpages of YouTube (80 kbytes on average) and Twitter (39.1 kbytes on average). YouTube showed the same behavior as Google, with almost no TCP problem, while the analysis with Twitter's experiments was similar to the one with Facebook. Table 6.9 reports the final results obtained in the four cases by applying our methodology. These results were expected because YouTube and Google are typically co-deployed, while Facebook and Twitter have similar webpage structures. They further confirm that our methodology yields consistent results across different experimental conditions.

Regardless of the service, country, or operator, we have shown that our automated anomaly detection system works and is functional to improve the networking service quality. Indeed, the system can self-identify network performance issues. So it can be used to alert the right support personnel, either TCP or radio experts in the presented example, which will receive as input the rules from, e.g., Table 6.5, that raised the alarm. Our STrees methodology is easy to implement, and it can be deployed in production environments.

Table 6.9: The final STrees decision tree performance with MONROE data-sets for YouTube and Twitter experiments.

Services	**Country**	**Operators**	**Accuracy (%)**	**Precision (%)**	**Recall (%)**	**F1 Score**	**Radio Problem**	**TCP Problem**	**Prob in both**
YouTube	**Sweden**	*op*0_sw	98.8	98.9	98.9	98.9	121200	2340	122
		*op*1_sw	98.8	98.9	98.9	98.9	92473	897	449
		*op*2_sw	99.1	99.2	99.2	99.2	80054	4280	1203
	Norway	*op*0_no	99.6	99.6	99.6	99.6	171094	534	1814
		*op*1_no	99.3	99.3	99.3	99.3	163712	234	7567
		*op*2_no	99.5	99.5	99.5	99.5	61605	1960	26
	Italy	*op*0_it	99.3	99.3	99.3	99.3	47544	28	268
		*op*1_it	99.5	99.5	99.5	99.5	22503	3071	1071
		*op*2_it	97.9	98.0	98.0	98.0	18266	2155	804
	Spain	*op*0_es	98.7	98.5	98.9	98.9	80248	2	20
		*op*1_es	98.7	98.9	99.0	99.0	28229	1884	2951
		*op*2_es	99.0	99.7	99.7	99.7	22430	840	1020
Twitter	**Sweden**	*op*0_sw	98.8	98.9	98.9	98.9	23591	15165	6565
		*op*1_sw	98.8	98.9	98.9	98.9	24907	7442	2653
		*op*2_sw	99.1	99.2	99.2	99.2	37615	11108	18926
	Norway	*op*0_no	98.8	98.7	98.7	98.7	66112	4306	6803
		*op*1_no	98.3	98.3	98.3	98.3	34017	14908	19618
		*op*2_no	98.7	98.7	98.7	98.7	17406	23543	28127
	Italy	*op*0_it	99.3	99.5	99.5	99.5	27306	3051	2894
		*op*1_it	99.7	99.7	99.7	99.7	14679	7584	2674
		*op*2_it	98.4	99.2	99.2	99.2	17378	14	6
	Spain	*op*0_es	96.7	96.7	96.6	96.6	19041	8280	7896
		*op*1_es	98.9	98.8	98.8	98.8	5403	1364	987
		*op*2_es	99.0	98.8	99.8	99.8	8459	2572	4288

6.4. Discussion

In this chapter we have presented a supervised ML methodology that allows to fully automate the process of identifying anomalies in the behavior of a network. The main advantages of STrees are that it can be implemented with limited supervision, and that its results are interpretable by humans.

We have also provided a few application examples based on real data from operational cellular networks. Notwithstanding the data-sets were obtained under very heterogeneous conditions and from very different networks, we have shown that we can easily identify behavioral anomalies (specifically, we have dealt with the case of TCP throughput, although the methodology is not tied to TCP) and that we are also able to further investigate and identify their root causes. This feature is key to promptly activate precise and effective troubleshooting actions.

Furthermore, our methodology is generic and can be used to automatically detect several types of networking problems, and examine several classes of potential anomaly causes, without losing the interpretation of the results. As such, it can be stationed by current cellular networks and used for future cellular monitoring

7 Causality Inference of Anomalies in Networks

Chapter 6 showcased that it is possible to build a troubleshooting tool to detect and classify networking anomalies using interpretable ML algorithms. However, one of the drawbacks of that approach is the continuous need for an expert in cellular networks to select the best KPIs to group the networking problems. Moreover, STrees uses supervised and unsupervised ML, and does not take advantage of all the features available in the data-set. To make troubleshooting even more automatizable, this Chapter presents an enhanced methodology which makes use of all available features in the data-set under observations and employs unsupervised feature selection techniques. We also eliminate the use of hyper-parameters and automate the complete process.

One can argue that it is possible to resort to Explainable AI, or XAI, instead of using simple ML algorthims which deals with the problem of how human users could understand AI's cognition, and decide if an AI based model can be trusted or not [21]. However, XAI does not help interpret AI decisions and conclusions *per se*, although its transparency of operation and interpretability of decisions allows humans to identify which features of the observed system led to specific algorithm's outputs on a case-by-case basis [93]. Multiple methods have been proposed to address the complex issue of ML interpretability, from determining which features contribute the most to a neural network's output, to the development of targeted models that explain individual predictions [22]. However, other studies point out that there is a potential need for applying an interdisciplinary joint efforts in order to correctly apply and leverage explainable/interpretable ML algorithms [94].

In this Chapter, we focus on anomalies as to situations that should not happen, given the capacity of the network and the current network traffic. Anomalies are not the same as problems. For instance, while an overloaded base station located in a football stadium full of people yields a network problem, that is not an anomaly. In fact, there is a clear reason for the base station to be very busy and provide low per-user throughput. On the contrary, an idle base station when the stadium is full is not a network problem (no network alarm is raised), but it is an anomaly, since the base station should be busy. This situation should raise an alarm and call for examination in order to seek for the causes of this anomaly, which might go beyond the network. We are interested in the identification of anomalies because they complement other tools for the identification of problems across the components used to establish a communication between users and server in a network. Moreover, anomalies are, hopefully, easier to solve than network problems, since they deal with an abnormal operation of the network, not with its limitations.

To perform anomaly detection and classification of their causes, here we only resort to simple and unsupervised ML tools to process the data collected from the network. We therefore develop an upgraded and generalized version of STrees, which we call Causality Inference of Anomalies in Networks (CIAN). In addition, we provide a software implementation of the methodology named Troubleshooting Trees (TTrees). We do so by making the methodology fully unsupervised and adding ML techniques such as discretizing the target variable and feature selection. The interpretability is maintained intact, with a key advantage that CIAN does not require access to large volumes of training data.

The key contribution of this Chapter consists in proposing an unsupervised methodology for the detection and classification of network performance anomalies. Here we resort to Kolmogorov complexity and information theory concepts and tools to build ML anomaly classifiers automatically and with limited data samples. Hence, with this work, we join the ML research stream in networking, while focusing on the troubleshooting of possible network issues even with small volumes of measurement data. However, differently from existing ML proposals, we detect anomalies by simply identifying the scenarios that ML algorithms *cannot* learn/explain. For instance, a network that shows low throughput is not anomalous if its low performance can be explained, e.g., by detecting the presence of a bad radio link. For this reason, a novel aspect of our work is the use of simple interpretable ML algorithms, moving away from the current trend of high-accuracy ML algorithms (e.g., deep learning) that do not allow interpretation (and hence understanding) of their outcome. Specifically, we use decision trees in our methodology since they are transparent to inspection, hence interpretable [95]. In fact, having low accuracy ML models in not a problem to our system, since we consider as interesting (anomalous) the scenarios that are misclassified by the ML algorithms, and we do not want to miss them by overfitting. Then, the anomalous scenarios are classified by the potential causes of their behavior. Understanding and classifying these anomalous scenarios allow alerting the appropriate department to take corrective actions.

In order to evaluate the methodology, we use real operational network data. Two of the data-sets used here were collected for cellular service auditing purposes by Nokia in various European countries. They include thousands of features (i.e., operational parameter reports and statistics) and a few KPIs used to mark the quality of the network in each test (i.e., throughput, connection establishment time, download time, etc.). However, the available data-sets are also quite limited in terms of number of data samples, e.g., they only report hundreds or a few thousands of experiments each. Thus, these data-sets are not suitable for commonly adopted deep neural networks and ML methods requiring complex training. The other data-sets have been obtained in measurement campaigns conducted by us with MONROE, an open-access platform for multi-homed experimentation with commercial mobile broadband networks across Europe [96]. With MONROE, we have access to larger data-sets, although with a reduced number of features with respect to Nokia's data-sets. Moreover, while Nokia's data-sets report mostly TCP traffic, with MONROE we were able to experiment with TCP and QUIC. Furthermore, We analyze two datasets with imbalanced anomalies and test an alternative implementation of CIAN with a new clustering algorithm. We use this alternative implementation to draw performance comparisons that assess the shortcommings of TTrees and propose possible solutions to the latter. As a result, in this chapter we show how CIAN can be flexibly used to study heterogeneous and diverse data-sets to identify the root causes of anomalies without human intervention.

The experts in detection and classification of network issues from the research team have been instrumental in the validation of the methodology and the TTrees system. They have manually inspected the outcome of the unsupervised process, verifying that the scenarios that were classified as anomalous did in fact show strange combinations of observed features and KPIs, and that their unsupervised classification into *network aspects* (i.e., combination of observed operational features) that identify the causes of the anomalies was consistent. Additionally, the experts guided the development of a supervised implementation of the methodology, STrees, in which features were manually aggregated into a few relevant network aspects based on the experts' knowledge and experience, like it happens in currently implemented troubleshooting protocols. The comparison of the outcomes of unsupervised and supervised methods with the same data has also been used to validate the methodology.

As mentioned, we provide a Python open-source implementation of our methodology based on the Scikit-learn library [91] and a new library that we have implemented.[1]

The rest of the Chapter is organized as follows. Section 7.1 overviews the unsupervised methodology proposed. Section 7.2 explains in detail how this methodology has been impemented as the TTrees system. Section 7.3 presents and analyzes the results CIAN achieves when it is applied to real data-sets of various kind. Finally, we discuss the lesson learnt in Section 7.4.

7.1. Overview of CIAN

In this section, we briefly describe the framework, and the steps into which the automated unsupervised methodological process we propose is divided, without entering in the details of design and implementation, which will be instead the object of Section 7.2. We begin by defining some basic notations and concepts that will be used to describe the methodology. The basic notation used is summarized in Table 7.1.

[1] https://github.com/Mohmoulay/WoWMoM2021

Table 7.1: Basic notation.

Symbol	Description
M	Model
D	data-set
$L(M)$	Model length
$L(D\|M)$	Length of the data-set D given the model M
$\mathbf{x}$	Collection of data samples
$\mathbf{x}_i$	A specific data sample
x_{ik}	A specific feature of a data sample
$\mathbf{y}$	Set of target KPIs
m	Number of classes resulting from KPI discretization
C_1	Initial classifier (Knowledge Tree)
A	Set of anomalous scenarios
$\mathbf{x}^A$	Set of features in anomalous scenarios
$\mathbf{y}^A$	Set of target KPIs in anomalous scenarios
R	Set of features that are highly informative about the target KPI
r_a	Subset of relevant features for problem classification
α	Number of subsets selected for problem classification
C_2	Second classifier (Aspect Classification Tree)

7.1.1. The Core Idea for Detecting Anomalies

Our methodology has been developed to find and interpret anomalies in network communications, in an automatizable way. The core idea behind our approach has been inspired by the concepts of *Kolmogorov complexity*, and in particular, by the *minimum description length* (MDL) and *minimum message length* principles [97, 98]. According to these concepts, learning from data is equivalent to find regularities, or equivalently, compress data, i.e., describe data in the shortest possible way without loosing information. In particular, the MDL principle claims that the best model M for a data-set D is the one that minimizes the following expression.

$$L(M) + L(D|M), \tag{7.1}$$

where $L(M)$ is the length of a model M encoded as a string of symbols, and $L(D|M)$ is the length of the data-set given the model. Intuitively, this latter term can be seen as the length of the list of errors made by the model M when predicting KPIs from features with a specific data-set D, encoded using an optimal length code. In this sense, we divide our data-set into two disjoint groups: the compressible part of D that can be described through M, and the incompressible part that includes all the points in D that are wrongly predicted (e.g., misclassified) by M. In a network data context, the compressible part of D is therefore what the model M identifies as regularities in the network described with a given set of features and KPIs, while the incompressible part corresponds to unexpected behaviors for which the KPIs cannot be predicted based on the observed features. In general, the more complex we allow the model to be, the more regularities we will be able to find, but also the longer its representation will be; thus, the appropriate balance between model complexity and model accuracy has to be found, so as to minimize expression (7.1). In this paper we address this trade-off between model complexity and model accuracy by means of applying an incremental procedure over model hyper-parameters to find the most complex possible model that does not overfit the data-set. The particular families of models used are selected for their accuracy and interpretability, i.e., they produce models that identify the compressible part of D based on interpretable rules. From here it follows the claim that the incompressible part of D cannot be interpreted, and hence it contains *anomalies*.

7.1.2. Input

The input to the process is a collection $\mathbf{x}$ of data samples, obtained from n network operation scenarios. Each data sample $\mathbf{x}_i$ includes k features $(x_{i1}, \ldots, x_{ik})$, which are operational characteristics of the network and the context in which measurements are taken, in scenario $i \in [1, n]$. In addition to these features, there is a distinguished set of ℓ target KPIs $\mathbf{y}$ which drives the search for anomalies. Hence, the data-set is formed by $\mathbf{x}$ and $\mathbf{y}$, and each scenario $i \in [1, n]$ is a row of the data-set containing features $(x_{i1}, \ldots, x_{ik})$ and KPIs $(y_{i1}, \ldots, y_{i\ell})$. Note that realistic data-sets can contain incomplete data entries. Therefore, before processing further a data-set, we will possibly filter out malformed or incomplete rows, and all rows that do not contain values for the target KPIs.

7.1.3. Discretization

The first step of our methodology consists in grouping data samples into classes, based on the value of the target KPIs $\mathbf{y}$. The reason for having this step is because, in general, the target KPI values belong to an infinite domain.[2] Note that our methodology is oblivious to the process that collects data samples, as we simply take a data-set and work with all its valid entries. For discretization, it is desirable that the number of classes m used is manageable but not too small, for expressiveness. However, we do not need vast amounts of data, as our methodology works with as few as tens of samples per discretization class, so as to have a bare minimum level of statistical relevance. Additionally, it is desirable that data samples are reasonably balanced across classes, although this may not be perfectly achievable in all cases. Finally, in order to provide semantics to the discretization, every vector of KPIs $(y_{i1}, \ldots, y_{i\ell})$ is assigned a certain QoS, which can be quantified. Then, the discretization places in the the same class scenarios whose QoS are similar, and the m classes can be ranked by an expert based on their goodness (e.g., as “very bad", “bad", “medium", “good", “very good", etc.).

7.1.4. Selection of Anomalous Scenarios

Once the data samples have been assigned to the classes based on the value of the target KPIs, we use them to train a classifier C_1. This classifier will use the features $(x_{i1}, \ldots, x_{ik})$ to determine whether, in scenario i, the KPIs $(y_{i1}, \ldots, y_{i\ell})$ seem to belong to class $j \in [1, m]$. The objective is to build a classifier that is able to correctly classify as many data samples as possible without overfitting. It is also desirable that the decisions of the classifier can be interpreted.

Applying this classifier C_1 to all the data samples will hence incorrectly classify a subset of them, out of which we consider the subset $A \subset [1, n]$ for which the classifier predicts classes higher than the observed ones. The scenarios in set A are the ones that we consider interesting, or *anomalous*: scenarios whose target KPIs is overestimated with

[2] We explored options without discretization, but they were unsatisfactory, because they led to complex approaches with additional hyperparameters.

a properly trained classifier C_1.[3] The interpretability of the classifier helps to determine why the data samples in A are anomalous (with respect to those properly classified). This means identifying which features are relevant and how to differentiate A from the rest.

Observe that the scenarios in A are not necessarily those in which any of the target KPIs show low performance. For instance, if $\mathbf{y}$ consists of a single KPI (e.g., the average throughput observed), a scenario i with low throughput y_i may not be anomalous if this is due to a low radio coverage (i.e., when the user is at the cell edge). This scenario should be correctly classified by an interpretable classifier C_1 as bad, and a visual inspection of the radio features in $\mathbf{x}_i$ should reveal the issue. The anomalous scenarios that are of interest to us are those that cannot be (directly) explained. For instance, a scenario i with low throughput y_i in which the features $\mathbf{x}_i$ are all good.

Hence, from this point on, our target will be to identify automatically which types of anomalies the scenarios in A show. This could later be used to report the issue (complemented with the data samples) to the appropriate department, that may take corrective action.

7.1.5. Selection of the Most Relevant Features

Let $\mathbf{x}^A$ and $\mathbf{y}^A$ be the set of features and corresponding KPIs in anomalous scenarios, taken as $\mathbf{x}_i$ and $\mathbf{y}_i$, for $i \in A$. These are the scenarios that we want to explore further. A natural approach to attempt for understanding what makes these scenarios anomalous would be using another classifier only for them. While this indeed separates these scenarios into different classes, our experience has shown that the results are however hard to interpret, even by an expert.

For this reason, for the sake of interpretability, in our methodology we proceed by restricting the set of features for further analysis. Hence, in the next step of the

[3] In some practical cases, it may be interesting to consider in the set A also the scenarios for which the classification assigns a worse class (in QoS terms) than their actual class. Anomalies that have a positive impact on QoS are not harmful at all for the user and could be treated less urgently than anomalies with negative impact on QoS, so we ignore them in this work. Still, they would deserve further studies because they could reveal the existence, and possibly explain the cause, of new interesting effects to be considered for design and tuning of network systems and services. The methodology described in this article can be straightforwardly extended to analyze also this kind of anomalies.

methodology, we select a small subset $R \subset [1,k]$ of features. The set R should contain only non-redundant features, and should contain those that are highly informative with respect to the set $\mathbf{y}^A$. The process we apply to select R is as follows. First, rank the features of $\mathbf{x}^A$ by the amount of information they provide about $\mathbf{y}^A$. Then, using this ranking, select the top features in order, removing those that are redundant with respect to the previously selected features. This process stops when a certain number of features have been selected, or when the next feature in the ranking gives little information about $\mathbf{y}^A$.

7.1.6. Clustering Using the Most Relevant Features

The next step of the methodology is to use the features in R to classify the anomalous scenarios into meaningful types. We have observed empirically that the features in R naturaly belong to different network aspects (e.g., radio features, TCP features, QUIC features, etc.), and that an anomalous scenario may show issues in several of these aspects simultaneously. This leads to a process in which subsets $r \subseteq R$ are selected, and each subset is used to classify the anomalous scenarios using binary clustering indicating whether the selected subset of features seems to be (at least partially) responsible for the anomaly. The conjecture is that this (unsupervised) process will select subsets that correspond to different network aspects, and the clustering will separate "good" scenarios from those with issues in that aspect. While this process is unsupervised, our experience is that it leads to subsets of R that an expert can easily match with particular aspects of the network.

Hence, in this step we select α subsets $r_a \subseteq R$, $a \in [1,\alpha]$, and apply each of them to divide the set A of anomalous scenarios into two clusters, a *low-performing* cluster (or a cluster with issues with respect to that network aspect, indicated with a bit 0) and a *high-performing* cluster (bit 1). The bit labels 0 and 1 are assigned automatically to the clusters, depending on which of them has better performance (in terms of QoS). Moreover, it is desirable to have balanced, compact and non-redundant clusters. Each anomalous scenario will be assigned to one of the two clusters generated with each subset

r_a. This can be represented as a binary string of length α, with a value 0 or 1 in each position, which classifies all anomalous scenarios A into 2^α types. Each of the bits in the vector associated to a scenario conveys whether that scenario shows issues in the particular network aspect.

7.1.7. Aspect Classification of Scenarios

Finally, a second classifier C_2 is built using $\mathbf{x}^A$ and $\mathbf{y}^A$ as training data, and the 2^α types as the classes to which these scenarios are assigned. Observe that we do not restrict the classifier C_2 to use only the features from R (although it is very likely that they will be used). Thus, C_2 uses for training $\mathbf{x}^A$, $\mathbf{y}^A$, and the α-bit string of every scenario in A (identifying the presence of anomalies in the α selected network aspects, as described in Section 7.1.6). The output of C_2 will group anomalies into 2^α classes, each with a specific anomaly pattern.

It is important that this classifier is interpretable, because this makes experts able to understand why an anomalous scenario has to be considered to have issues or not in each of the automatically identified network aspects. Note that if an expert had been involved in the most relevant feature selection process, she should have assigned a specific network aspect to each subset $r_a, a \in [1, \alpha]$, and possibly a meaning to every binary value of every element of the network aspect anomaly string. Be it the output of manual inspection or automatic data analysis, this information can be used to determine the issues each particular anomalous scenario has, and can be sent to the appropriate department to promptly intervene.

7.1.8. Using the Classifiers to Detect and Identify Anomalies

After the previously described steps have been completed, we found ourselves with two classifiers C_1 and C_2, that can be used to detect anomalous scenarios, determine network aspects that make this scenario anomalous, and hence notify about this to the appropriate department for troubleshooting. Now we can use these trained classifiers to analyze fresh data inputs and detect anomalies. In particular, consider a new scenario

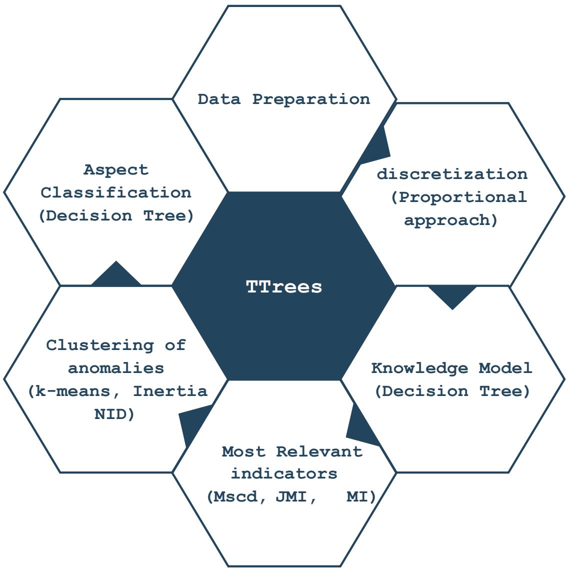

Data Preparation
discretization (Proportional approach)
Knowledge Model (Decision Tree)
Most Relevant indicators (Mscd, JMI, MI)
Clustering of anomalies (k-means, Inertia NID)
Aspect Classification (Decision Tree)
TTrees

Figure 7.1: Steps of TTrees (using multiple ML techniques): beginning with data preparation, proportional discretization, training of knowledge tree, selection of the most relevant features, identification of anomaly clusters, and training of a network aspect anomaly classifier.

$(x_1, \dots, x_k, y_1, \dots, y_\ell)$. Using classifier C_1 we determine that this is an anomalous scenario if the KPI class C_1 assigned to $(x_1, \dots, x_k)$ is not the one corresponding to the discretization class of $(y_1, \dots, y_\ell)$ but it is actually better. If that is the case, C_2 will be used, so that the class assigned to $(x_1, \dots, x_k, y_1, \dots, y_\ell)$ is an α-bit string expressing whether this scenario has issues in each of the α different network aspects or not, which allows for alerting the appropriate department(s).

7.2. Implementation of the CIAN methodology

Here we present the implementation details of the tool TTrees that automatizes CIAN, the methodology presented in the previous section. Fig. 7.1 depicts the different phases of the proposed methodology according to their implementation in TTrees. As reported in the figure, our implementation leverages a number of standard tools for data analysis and ML, from discretization algorithms to clustering and classification.

7.2.1. Data Preparation in TTrees

The first phase consists in preparing the available data. This is a necessarily preliminary step that includes data filtering, cleaning of the data-set from the presence of incomplete entries, and extracting the k features in $\mathbf{x}$ and the target KPIs in $\mathbf{y}$ that need to be observed (i.e., note that the given data-set might include more features and KPIs than what we are interested in). Thus, we use a Python script to extract $\mathbf{x}$ and $\mathbf{y}$ from the data-set. The script automatically discards entries in which one or more KPIs or features are not reported. It also discards features that are constant through the data-set, which would not be informative. Features can be either numeric values or categorical entries (e.g., labels used to indicate the operational conditions of the network, like the name of the radio technology used, the name of the protocol adopted for transmission, etc.).

7.2.2. Discretization in TTrees

The selected target $\mathbf{y}$ is usually a set of one or more numerical KPIs, whose values may lay on a continuous real interval. Thus, a first problem to address in the training of TTrees consists in discretizing the continuous target into a finite number of classes (i.e., categories). Using a simple quantization of the values in $\mathbf{y}$ would introduce an error in the subsequent data analysis. Hence, instead we keep the target values as they are in the data-set, and assign a label to each point, which identifies their class. Since TTrees is unsupervised, discretization uses progressive numbers as labels (e.g., “0” to “5”).

In our methodology, being based on the study of the compressibility of the training data-set using optimal lengths codes, it is of upmost importance that the discretization algorithm applied does not alter the distribution of the training samples, that is, both the continuos KPI and the discretized version should follow the same distribution. A uniform discretization is an easy to implement approach that satisfies this requirement. Therefore, we adopt a uniform discretization of target KPIs.

The number of classes (labels) used is not pre-defined. Instead, it is automatically derived by TTrees from the available number of points in the data-set. There exist a collection of candidate discretization algorithms that are not biased and have low variance (e.g., equal width, equal frequency, or fixed-frequency). Unfortunately, they require optimizing hyperparameters [99], which makes them inadequate for an unsupervised automated software tool. Instead, we use a discretization approach that does not require any hyperparameter tuning: *proportional discretization* [100]. This method uses a number of categories proportional to the size of the data-set. The number of categories used, denoted by m, is computed as $m = (\log_2 n)/2$, where n is the number of samples of KPIs $\mathbf{y}$. We use the proportional discretization approach to identify the centroids of the m intervals of KPI values (associated to the m categories). After identifying the centroids, we apply k-means clustering to the continuous target variable $\mathbf{y}$, which assigns labels to the data points and returns the boundaries between categories intervals.

7.2.3. Knowledge Model for Identifying Anomalies

Once the data points $\mathbf{y}$ are assigned to a finite number of categories, TTrees continues by building a knowledge model, meant to identify anomalies. For this step we use an ML classifier, which we want to be interpretable, so as to allow the user to understand what kind of anomaly is detected. This is achieved by training a decision tree [101] with the data set $\mathbf{x}$, and using as classes the categories defined in the discretization phase. The decision tree infers the class for a sample $\mathbf{y}_i$ of $\mathbf{y}$ from the values of the associated features $\mathbf{x}_i$. This justifies the use of terms "knowledge model" and "knowledge tree" for the role of this classifier. However, some samples $\mathbf{y}_i$ can be misclassified, representing anomalous behavioral trends that cannon be learned. Note that, with more advanced models (neural networks, random forests, etc.) instead of a decission tree, we could gain accuracy, but we would lose interpretability and the model may discard interesting samples.

The knowledge tree is build by applying the widely adopted Classification And Regression Tree (CART) algorithm [102], with the Gini impurity metric [103] computed by comparing the output of the classification with respect to the discretization categories. We refer the resulting decision tree as C_1. The depth of C_1 is limited to a maximum of $\lfloor(\log_2 n)/2\rfloor$, so that, on average, the leaves of the tree hold a sufficiently large number of samples to properly classify the KPIs. A deeper tree would suffer a high risk of overfitting [104]. Moreover, we also limit the number of samples at the tree leaves to a minimum of five, to further reduce the risk of overfitting [103].

After the full tree has been derived to the maximum depth allowed, we perform a cost-complexity pruning, which is the best known approach to avoid overfitting in potentially large trees [103], combined with cross-validation of the candidate pruning points. The leaves of the tree that have the highest Gini impurity measure are removed first. Pruning minimizes a cost-complexity metric that linearly combines the classification error (cost) made by the tree and the size of the tree (complexity). However, the pruning metric has a hyperparameter which represents the relative importance of cost with respect to complexity. Cross-validation is used to tune that hyperparameter automatically. In practice, for each candidate value of the hyperparameter, we identify the tree with the minimal cost-complexity, and cross-validate the performance of that tree by splitting the data-set into smaller pieces of at least 30 samples each: we use all but one of the pieces to train the tree and the remaining one for evaluation. This process is repeated as many times as the number of pieces in which we break the original data-set (so as to use every piece for validation exactly once).

Once C_1 is created, for each sample of $\mathbf{y}$ we have two possibilities: the sample is classified or not under the same class in the discretization phase and by the knowledge tree. If the knowledge tree returns a class higher than the one of the discretization phase, an anomaly of interest is identified and included in set A.

7.2.4. Most Relevant TTrees Features

The next step consists in identifying a subset of features that are relevant for the anomalous scenarios in A, so as to identify which network aspects are relevant to explain the anomaly. TTrees then evaluates each of the available features by computing how much information on the target $\mathbf{y}$ is contained in the feature. For this task we use the well known metrics *mutual information* (MI) [105] and *joint mutual information* (JMI) [106], from classic information theory.

We have also tested a novel metric that we have built on top of the concepts of

Kolmogorov complexity and normalized compression distance (NCD) [107], which we call *miscoding* (Mscd). The miscoding of a sequence of p different features $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, \dots \mathbf{x}_p\}$ with respect to the target $\mathbf{y}$ is defined as

$$\mathrm{Mscd}(\mathbf{x}_i, \mathbf{y}) = \frac{1 - \mathrm{NCD}(\mathbf{x}_i, \mathbf{y})}{\sum_{j=1}^{p} (1 - \mathrm{NCD}(\mathbf{x}_j, \mathbf{y}))}. \tag{7.2}$$

We propose Mscd because it measures the difficulty of reconstructing the target $\mathbf{y}$ from the features of $\mathbf{x}$ and vice versa, which accounts for the redundancy in $\mathbf{x}$. In TTrees we start by computing a conditional redundancy matrix with the normalized compression distance of all possible pairs of features with respect to the target variable $\mathrm{NCD}(\langle \mathbf{x}_i, \mathbf{x}_j \rangle, \mathbf{y})$, based on a joint discretization of the attributes, and the computation of optimal length codes, given the relative frequencies of the discretized vectors. Then, we select the attribute with the highest NCD, and recompute the values of the redundancy matrix assuming that this value is selected. We repeat this process of selection of the best feature and recalculation of the redundancy matrix until all the features have been selected. The final miscoding is given by normalizing over one minus the NCDs computed for the different attributes.

MI is applied on a per-feature basis and is able to quantify the relevance of a feature, but it fails to identify redundancy. JMI is instead used on feature pairs, so that it allows to evaluate not only the relevance of an feature but also the redundancy with another feature. However, JMI is prone to errors in case outliers are present. Mscd offers the possibility of evaluating relevance and redundancy, plus the quantity of irrelevant information contained in the feature.

We will compare the performance obtained by applying MI, JMI and Mscd in the numerical evaluation section. Here it is enough to mention that in all of the three cases, we sort the features in decreasing order according to the selected metric, and pass the top of the list to the next TTrees phase. In particular, since we will need to cluster the anomalies based on selected network aspects, we pass to the clustering phase a number of features much larger than the number of aspects to study. For this paper we have used m^2 features, where m is the number of bins in which the target KPI has been discretized.

7.2.5. Clustering of Anomalies in TTrees

Each *pair* of relevant features is regarded as a potential *network aspect.* For each aspect, TTrees applies a clustering algorithm on all anomalous samples. This bidimensional clustering is done with the k-means algorithm [103], using the normalized values of the features obtained via RobustScaler [108]. In addition, and for visualization purposes only, when we have just one target KPI, we use a simple regression with respect to $\mathbf{y}$ on each of the features, and we sort samples according to increasing regression values. This allows to plot anomalous samples consistently so that higher $\mathbf{y}$ values are at the top-right corner of the plots. As a consequence, the cluster of samples at the top-right corner contains the samples with highest $\mathbf{y}$ values and the cluster at the bottom-left corner the samples with lowest $\mathbf{y}$ values. If the target KPI $\mathbf{y}$ is better when larger (resp., smaller), then the bottom-left (resp., top-right) cluster is the one with issues in the network aspect defined by the two features. In general, the cluster with issues is assigned a label 0 in this aspect, and the other cluster is assigned 1.

The problem is that we have potentially a large number of pairs of features (and hence network aspects), and some of them might bring redundant information. Therefore, TTrees selects only a few pairs of features as network aspects, so as to easily and interpretably identify non-redundant descriptions of anomalies. In our current implementation we select $\alpha = 4$ feature pairs, i.e., the 4 most relevant aspects to evaluate to understand the anomaly. Thus, the number of features passed from the previous step is $p = \alpha^2 = 16$. We do not use all possible features for a matter of practicality. For example, if we had $p = 120$ features, which is a reasonable value for cellular data traffic traces, we would have to evaluate $p(p-1) = 14280$ pairs, and make hundreds of millions of comparisons to test the redundancy among pairs.

The most relevant and descriptive feature pairs are selected based on multiple metrics. First of all, TTrees computes the inertia score [109] of the clusters resulting from each feature pair. The inertia of a cluster quantifies the tightness of the clusters (the lower, the better). The resulting ranked list is filtered by using two criteria: TTrees only keeps clusters with low redundancy and which are balanced in the number of elements in the two clusters. Redundancy of clusters is computed using the normalized information distance metric (NID) [110]. Here the order matters, since if two feature pairs yield redundant clusters, the feature pair with lower inertia is retained, while the other is discarded. In order to automatically tune and apply both filters, we select filtering thresholds with the help of the histograms obtained for the distributions of redundancy scores and balance metric. We have observed that the histogram of redundancy shows a bimodal distribution, therefore we select as threshold to accept/reject a feature pair the point with the lowest value in-between the two peaks of the bimodal distribution. For balancedness, the distribution histogram is multimodal, with an automatic threshold selection where the minimum point between first and second peaks (minimum accepted value of balancedness), and the minimum point between the last two peaks (maximum accepted value for balancedness).

7.2.6. Aspect Classification in TTrees

The α top feature pairs, obtained in the previous phase of TTrees, identify network aspects in which the anomaly could be rooted. In this phase they are used to train an interpretable classifier (again a decision tree) to map the anomalies A onto the 2^α classes corresponding to the combinations of the α network aspects relevant for the anomalies. This *aspect classifier*, denoted as C_2, uses the the binary values of the aspects assigned in the previous phase, and builds new categorical target variables consisting of binary strings that combine these binary aspect values. The elements with value 1 in the binary string identify aspects in which the scenarios of that category may have issues, and for which a network specialist should be called. Note that this classifier makes decisions based on the full list of features, and not only based on the most relevant features identified by TTrees. In the training of C_2, like for C_1, we avoid overfitting by limiting the depth of the tree to $\lfloor (\log_2 |A|)/2 \rfloor$, and apply cost/complexity pruning with cross validation to increase the generalization level of the tree.

7.2.7. Software Implementation

TTrees is implemented using Python. We use two libraries: (*i*) the widely adopted scikit-learn library, which provides us with ML algorithms, such as decision trees and k-means, and (*ii*) a library developed by us for automatic ML tools, which includes an Mscd calculator.

7.3. Empirical Evaluation

Existing troubleshooting tools do not generalize very well and need the assistance of an expert. For this reason, here we evaluate TTrees along with a supervised version of our tool, which provides a validation baseline. For simplicity, here we use only one target KPI (e.g., the throughput) and restrict the analysis of anomalies to the case in which the knowledge model predicts better throughput than what observed. We also present a modified version of TTrees, in which we modify the clustering algorithm in order to effectively analyze imbalanced data-sets. The first data-sets used in this section were extracted from experiments executed in controlled environments. These are complemented with Google and Facebook data-sets, which are the result of homogeneous cellular network conditions and include nodes deployed in both static and mobile stations. The latter can suffer from network outages and undergo unexpected variances in the quality of the signal, which enriches the possible scenarios captured in the data-set and can be a clear source of anomalies. As it will be later developed, these uncertain conditions yield aspect classification clusters where the number of anomalies is imbalanced, that is, the ratio is not equal between clusters and one class accumulates most of the samples.

The data-sets collected for this paper and the analysis of all data-sets presented here, for all services and operators discussed, are available in GitHub.[4]

[4] https://github.com/Mohmoulay/WoWMoM2021

7.3.1. Supervised ML-based Troubleshooting

The supervised version of TTrees, denoted as STrees, uses a supervised selection of network aspects that are most relevant to identify and cause anomalies in the problem classification phase of the methodology. In particular, with the supervision of a network expert, STrees selects as network aspects those combinations of features that are most relevant for the selected target KPIs, as identified by the expert. Afterward, in STrees, we take the tagged classes and use them to train C_2, i.e., the final aspect classification tree, like is done in TTrees.

Table 7.2: Brief description of the experimental data-sets used in this work

ID	Dataset	# Samples	# features	Families of features
#1	**HTTP FILE DL**	6,690	228	Radio, RTT, Duration, TCP Volume, TCP Flags, TCP Window, Packet Anomaly
#2	**HTTP LIVEPAGE DL**	10,500	126	Radio, RTT, Duration, TCP Volume, TCP Flags, TCP Window, Packet Anomaly
#3	**HTTP FILE DL MONROE**	120,000	106	Radio, RTT, TCP Window, Packet Anomaly
#4	**QUIC FILE DL MONROE**	3,951	91	RTT, Duration, TCP Volume, TCP Flags, TCP Window, Packet Anomaly
#5	**Facebook Page Download**	1447,839	109	Radio, RTT, TCP Window, TCP, Packet Anomaly
#6	**Google Page Download**	1261,119	109	Radio, RTT, TCP Window, TCP, Packet Anomaly

7.3.2. Datasets

Table 7.2 summarizes the characteristics of the data-sets used in this work. We first analyze two data collections of cellular network experiments used by Nokia in 2019 for network performance assessment of various European 4G networks (Datasets #1 and #2). These two data-sets contain drive-test information on TCP traffic experiments in which constant-size (3 MB) files are downloaded, and live web-pages are fetched with a mobile user device. Then, we further evaluate TTrees using larger mobile broadband connectivity data-sets collected with MONROE [111]. This includes generating a data-set of QUIC *qlogs* by running experiments from MONROE nodes.

More in detail, the Nokia data-sets used for experimental validation are real test records generated by processing all the information provided by the testing equipment used in the drive-tests, and aggregating the information at test level (count, sum, min, max, average, percentiles, etc.). The resulting data-sets have a single row per test and hundreds of columns summarizing all the dimensions (i.e., context features like date, time, location, network element information, etc.), technology features (radio, TCP/IP, application, etc.) and the KPI related to that particular test. The data transmitted in the drive tests is synthetic and does not include customer's sensitive information, protected by the European Union General Data Protection Regulation (GDPR) or similar regulations (i.e., the data has been generated by a testing device and not by real users). Finally, potentially sensitive data, such as the identity of the network operator, has been anonymized. The Nokia data-sets utilized in this study are rich but not sufficiently large to, e.g., train a neural network. They contain 54 228 and 21 655 rows, respectively, with 1326 columns.

The third data-set we use (Dataset #3) is a MONROE data-set with 120 000 rows and 106 columns. We have obtained access to MONROE and replicated the experiments of the first Nokia data-set (i.e., with 3 MB HTTP file downloads) on a larger scale, although with a different number of features. Moreover, some of the MONROE data-set highlights include the ability to filter by test type, infrastructure, operator, vendor, and transmission technology.

The fourth data-set (Dataset #4) was also obtained by means of the MONROE platform with the main aim of capturing traffic different than TCP. The experiments conducted in this scenario consist in the exchange of data packets from a server we control to a client residing on a MONROE network probe. Packets are sent over QUIC. From a data transport point of view, the novelties introduced by QUIC can be roughly summarized as follows: QUIC offers a combination of TCP reliability and multiplexing capabilities with the speed provided by the UDP transport protocol, over which QUIC is built [39]. In the experiments, we tested multiple possible variants for the congestion control algorithm implemented by QUIC, those being BBR, COPA, New Reno and Cubic. We also conducted two classes of QUIC experiments with sequential and parallel file downloads, respectively. The server can utilize multiple streams to exchange data with the client over QUIC. The number of streams spawned ranged from 1 to 5 across the experiments conducted. To limit the duration of the experiments, each opened connection had a lifespan of 15 seconds, enough to evaluate the behaviour of congestion control algorithms over variable radio conditions. Client nodes located in Norway, Sweden and Spain were utilized to run the experiments. Two types of clients were identified according to their mobility: static and mobile nodes. Among the mobile nodes available in the MONROE platform, the experiments conducted involved those which were installed on Swedish buses. Introducing mobile nodes contributes to increase the heterogeneity of samples with scenarios were the quality of network connectivity can be significantly lower than in static cases.

The complexity of the QUIC protocol and its use of encryption can hinder the inspection of QUIC traffic to such an extent that a specific format called *qlog* has been defined to organize and collect the information captured during a data exchange [23]. A *qlog* file includes not only QUIC events, but also protocol events, each of which includes a timestamp, the type of event and the value associated as specified in [87]: the timestamp indicates the time at which the event was registered, a field describes the type of event associated to the record, and finally, the data recorded in included. These *qlogs* have been the primary source of information to produce the data that populates the fourth data-set, which emulates the format and features present in the Nokia data-set. Therefore by applying TTrees to QUIC qlogs, we test its flexibility to work with heterogeneous kinds of data without the need of tuning any hyperparameter manually.

Datasets #5 and #6 were specifically designed to evaluate the connectivity to websites that should offer high performance, so that possible network anomalies could be experienced mostly because of radio problems. In this way, we obtain data-sets in which radio and transport anomalies are strongly imbalanced. Specifically, Dataset #4 collects measurements run against the Facebook webpage. We obtained the measurements by means of multiple Facebook webpage downloads in various European countries, using MONROE probes. The average downloaded webpage size of Facebook was 72 757.3

bytes and, at each experiment run, we cleaned the cache to make sure that a new complete download occurs. In some countries, such as Sweden and Norway, we had access to mobile nodes mounted on public transport vehicles. Thus, the downloaded page statistics for those countries are a mixture of statistics for static and mobile cellular nodes, hence the slightly higher radio problems which is depicted later in this subsection. As for Italy and Spain, we only report results for static cellular nodes. We run similar measurements with MONROE and Tstat against the Google search engine webpage, which was used to build Dataset #6. In those experiments, We observed an average downloaded size of 17 368.8 bytes, following the same procedure as the Facebook service (repeating downloads, clearing the cache, etc.).

In what follows, we use the first Nokia data-set to validate CIAN/TTrees and showcase its troubleshooting capabilities for the case of HTTP file download operations over LTE networks. We only consider data for a single operator (we anonymize the operator's name for privacy purposes). The other data-sets are used to showcase the potentials of CIAN and pros and cons of how the methodology is implemented. In particular, the second Nokia data-set is for HTTP webpage download performed live, over LTE networks, and one single operator as well. The first MONROE data-set Dataset #3 is for HTTP file download over LTE networks, and with one single operator, with tests anonymously conducted from labs and public buses in various European countries; so it is similar to the first Nokia data-set, but with more entries, less features and much more geographical diversity. The MONROE QUIC data-set Dataset #4 is relatively small, with less than 4000 entries, and reports very different families of features with respect to the other data-sets. Datasets #5 and #6 are used to show that, in some cases, we need to trade off accuracy for interpretability of CIAN results.

7.3.3. TTrees in Action and its Validation

For the analysis of the HTTP file download data-set of Nokia, we choose the *throughput* as the desired target KPI. Since there are 6690 samples, the number of categories used for this specific case is $\lfloor(\log_2 6690)/2\rfloor = 6$. Fig. 7.2 illustrates the results obtained when

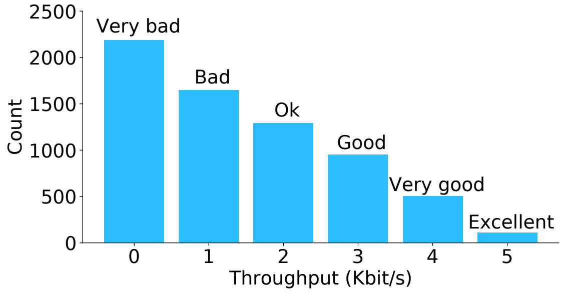


Figure 7.2: Target KPI (throughput) distribution for Dataset #1 via the proportional discretization approach

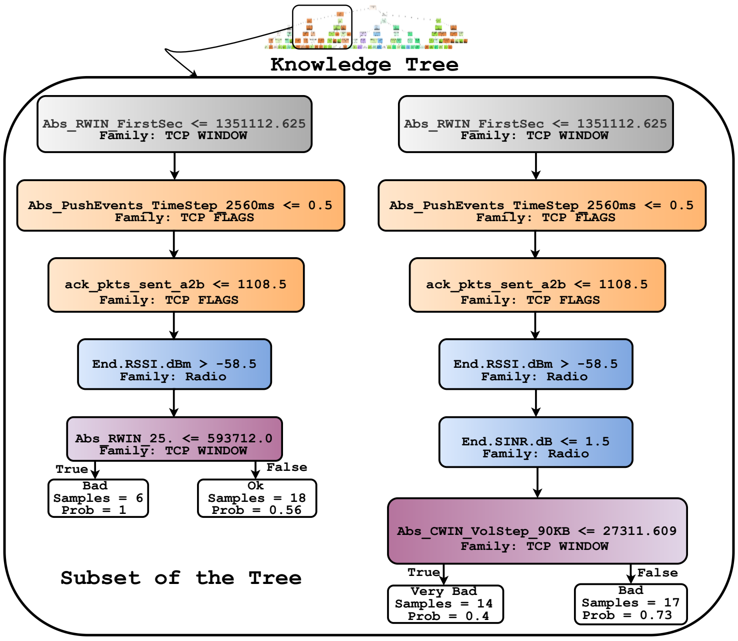


Figure 7.3: Zoom into a subset of the knowledge tree built for Dataset #1. The figure also reports which family of features the branching variable belongs to (see Table 7.2). Observe that both branches are the same up to the fourth node, after which the left corresponds to the *true* brach and the right to the *false* branch.

applying the proportional discretization approach to this KPI in TTrees. The discretized target variable is fed to the CART algorithm to build the knowledge tree. In CART, we fit the knowledge tree using all the 228 features available in the data-set.

Fig. 7.3 explains a subset of the resulting knowledge tree. For a given sample at the root of the node, the CART algorithm checks if the Abs_RWIN_FirstSec feature (the absolute value of the *TCP received window* during the first second of a connection) is less or equal to the self-learned threshold of about 1.3 MB. If this statement is true, the CART algorithm moves to the left side of the tree. Each node of the knowledge tree reports the value of Gini impurity, i.e., a measure of the fraction of misclassified data points that are

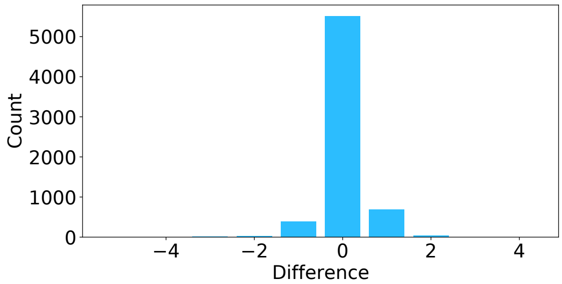
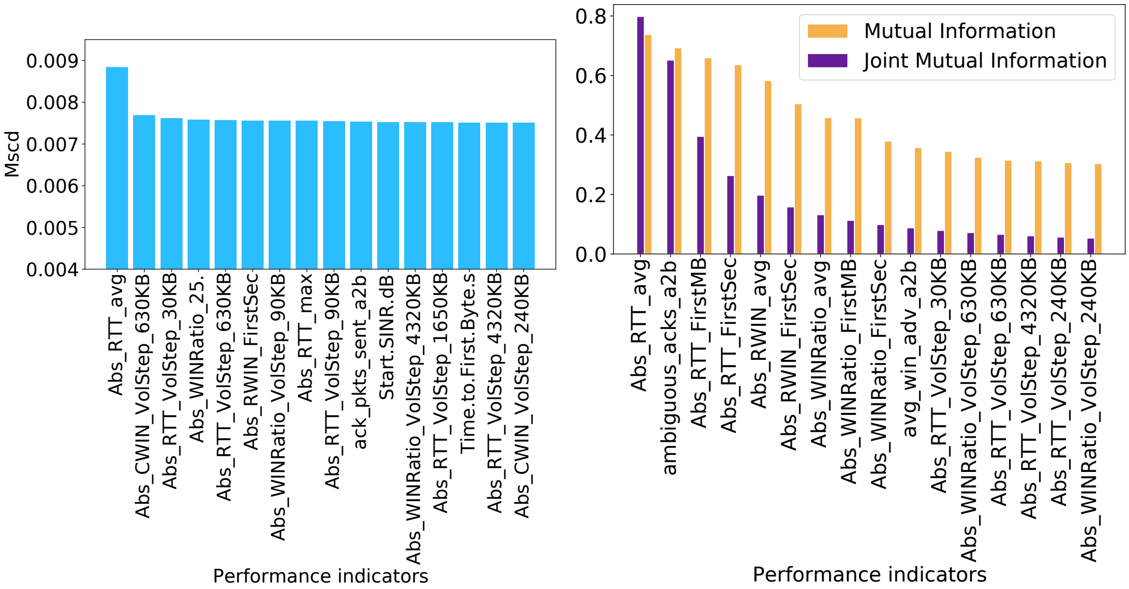


Figure 7.4: Difference (predicted - discretized) between the class assigned by the knowledge tree and the discretized category for the target KPI of Dataset #1

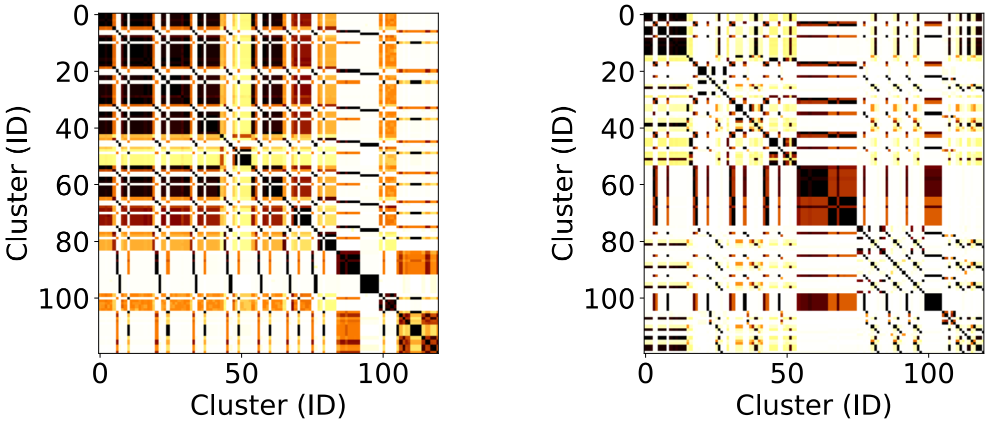


(a) Miscoding

(b) MI and JMI

Figure 7.5: Ranking of features according to their relevance to the description of anomalies for Dataset #1

reported in the tree leaves that can be reached from that node. It also reports the total number of samples examined in the node (6690 at the root) and the number of samples that corresponds to each discretization category. E.g., $[2185, 1646, 1293, 950, 506, 110]$ in the root node correspond to the number of samples that fall in the “Very bad” (0), “Bad” (1), “OK” (2), “Good” (3), “Very Good” (4), and “Excellent” (5) discretized throughput categories. The class value associated to each node is the one with the highest number of samples (“Very bad”, in the example). The full knowledge tree generates a meaningful set of classification rules, each based on a feature, and whose family (radio, TCP window, TCP flags, etc.) is reported in each node of the tree. That can help expert troubleshooters get an initial idea of what is going on in the cellular network, but since the full knowledge



(a) MI-based selection (b) JMI-based selection

Figure 7.6: NID heatmap matrix for cluster pairs. Darker colors represent redundancy.

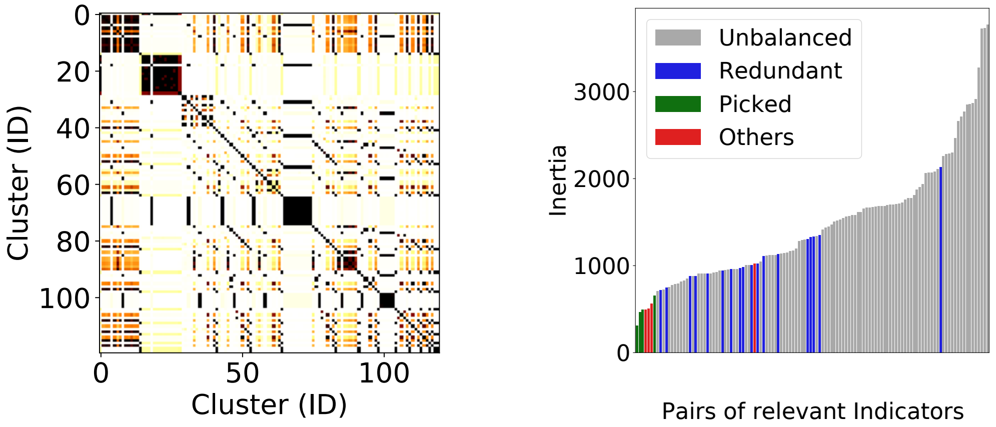
tree is wide, manual inspection and interpretation can be cumbersome.[5]

Fig. 7.4 shows that the knowledge tree classifies the target samples with high accuracy, although the number of cases in which the predicted class is better than the one observed in the discretization phase is not negligible ('-1' in the figure). TTrees selects these 740 anomalies to continue the analysis.

The number of network aspects that can be analyzed by composing binary clusters in the aspect classifier C_2 with a population of 740 anomalies is $\alpha = \lfloor(\log_2 740)/2\rfloor = 4$. Thus, the number of features to use is $\alpha^2 = 16$. Fig. 7.5 depicts the 16 most relevant features obtained by using the Mscd metric (left subplot) and by the MI and JMI metrics (right subplot). The figure clearly shows that MI and JMI practically find the same set of features, while Mscd identifies a significantly different subset, which is due to the fact that Mscd accounts for redundancy of features while MI and JMI do not.

In Figs. 7.6 and 7.7(a) the redundancy of the binary clusters obtained with the 16 most relevant features is shown, for MI, JMI, and Mscd metrics, respectively. The heatmaps shown in the figures represent the NID metric between cluster pairs: the darker the color the lower the distance between cluster pairs (and hence the higher their redundancy). Fig. 7.7(a) shows lighter colors than the other two heatmaps of Fig. 7.6, which means

[5] The full tree is large and cannot be correctly visualized here, so we made it available online at https://github.com/Mohmoulay/WoWMoM2021.



(a) NID of cluster pairs (features selected with Mscd)

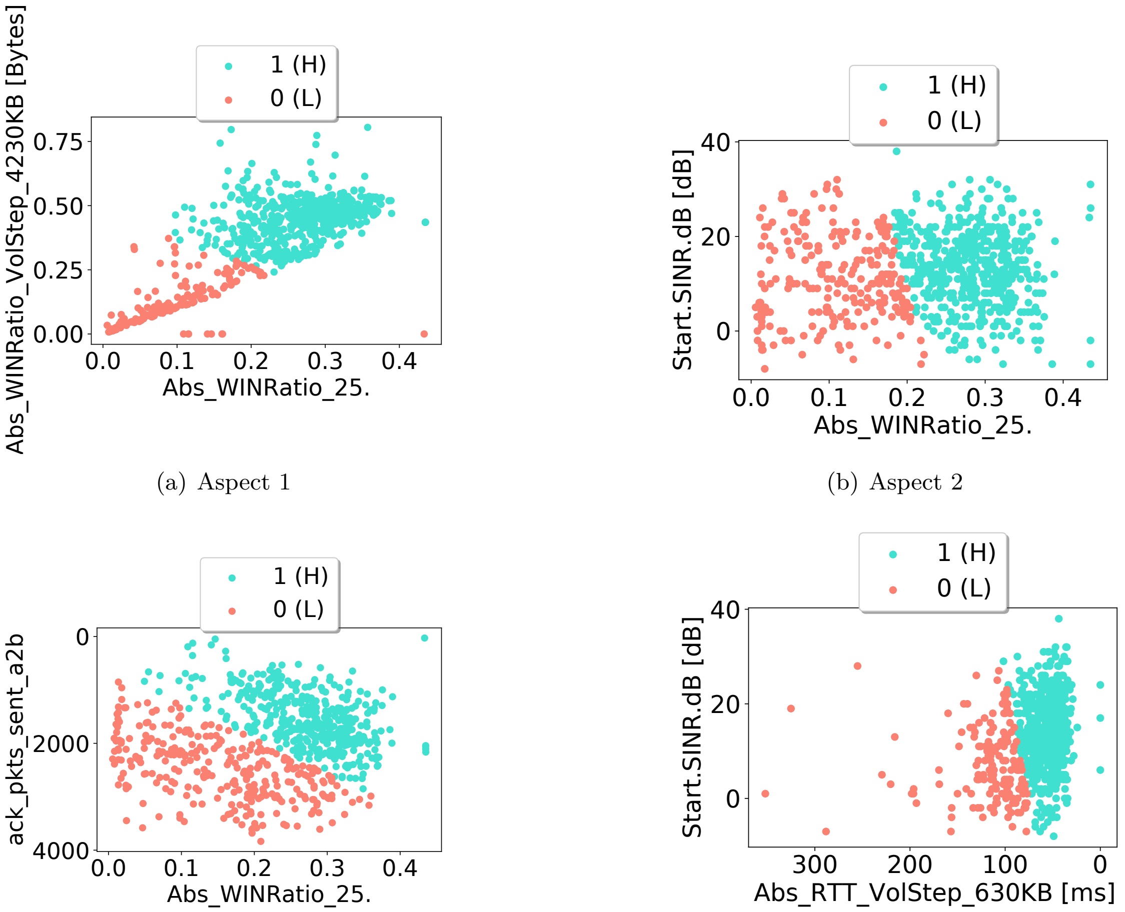
(b) Inertia of clusters (combinations of relevant features)

Figure 7.7: Network aspect selection based on miscoding; lower inertia of k-means clusters is preferred, unless clusters are redundant or imbalanced

that Mscd yields features whose combinations of clusters have lower redundancy. For this reason we will mainly consider the features selected with Mscd in the rest of this section.

Fig. 7.7(b) illustrates the inertia of the clusters obtained with the features selected with Mscd, and the filtering process based on balancedness and low redundancy criteria. The output of this filtering process is a set of 4 network aspects (in practice, 4 pairs of features) used for binary clustering. The visualization of the filtering results for the selection of relevant aspects obtained with MI and JMI is omitted because it is not relevant for comparison purposes. What matters is instead the set of results on the performance eventually obtained with those metrics, which will be commented later and which are reported in Table 7.3.

Fig. 7.8 shows the 4 network aspects obtained with the first Nokia data-set and the corresponding clusters to be used in the aspect classifier C_2. The figure shows that TTrees has identified anomaly clusters characterized by TCP operational parameters, radio parameters, delay parameters and packet anomaly. In STrees, instead, an expert would have to look manually at the “usual suspects” for the presence of anomalies, such as possible radio strength problems, TCP-related problem, packet anomaly and high RTT caused by buffering, and cluster them to see if they yield any groups. In this case, a Nokia cellular expert has identified for us the most relevant features, which led STrees to identify



(a) Aspect 1

(b) Aspect 2

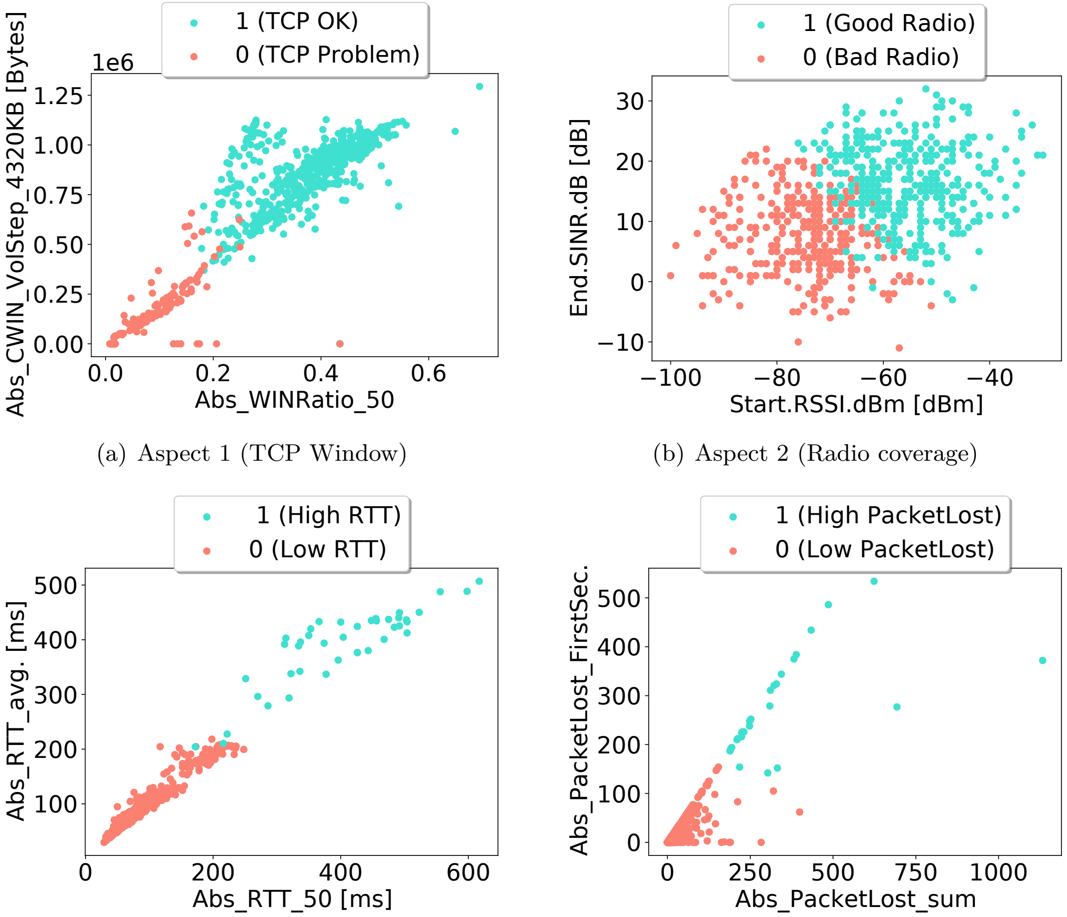
(c) Aspect 3

(d) Aspect 4

Figure 7.8: Aspects obtained with TTrees for Dataset #1 (using Mscd, see Figs. 7.5 and 7.7). The parameters used are Signal-to-interference-plus-noise ratio (SINR), Number of packet acknowledgment sent, absolute TCP Window ratio, and RTT.

the network aspects visualized in Fig. 7.9. Aspects identified by TTrees and STrees are quite different, except in both cases they cover the same families of features (radio, TCP window, RTT, packet anomaly). The actual features selected by TTrees could be less intuitive for an expert (e.g., selecting number of acknowledgment instead of packet loss), but without loss in information. In reality, for an expert, Aspect 3 from Fig. 7.8 makes sense but may not be the first (intuitive) choice; instead, he/she would first look at the packet loss from Fig. 7.9. Indeed, one of the advantages of the TTrees methodology over STrees is the capability of creating meaningful yet not intuitive inter-family aspects combining radio with TCP, TCP window with packet anomaly, etc.

The corresponding C_2 trees trained with TTrees and STrees are (partially) reported



(a) Aspect 1 (TCP Window)

(b) Aspect 2 (Radio coverage)

(c) Aspect 3 (RTT)

(d) Aspect 4 (Packet anomaly)

Figure 7.9: Aspects obtained with STree for Dataset #1

in Fig. 7.10. The same four families of identifiers are deemed as relevant by supervised and unsupervised approaches: Radio, TCP-based, RTT, Packet anomaly. TTrees detects a dependency in Aspect 4 that is negatively correlated: when the value of the specific feature Abs_RTT_VolStep_630KB (i.e., the RTT observed after averaging over chunks of 630 kB of transmitted data) is higher than 87.909 ms, the throughput is going to be lower. This means that channel capacity will be the limiting factor causing buffering and we have a negatively correlated relation in Aspect 4. Now, if we look at the right side of Fig. 7.10, STrees is describing low RTT and high packet loss as well, deriving that result from the same TCP features plus radio. Furthermore, if we look at the whole tree, they both convey similar results. In practice, TTrees and STrees are comparable and show similar classification, taking into account the difference seen in the aspect selected.

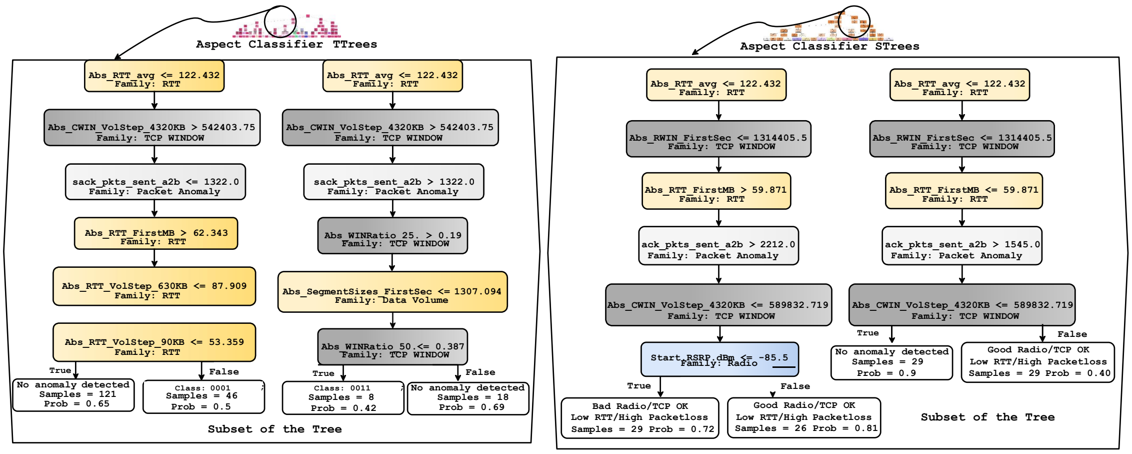


Figure 7.10: A comparison between the supervised and unsupervised approach showing the resulting aspect classifiers

For instance, the root node is the same: Abs_RTT_avg (i.e., the average RTT observed during file download) is less or equal to 122.432 ms, but the tree leafs can not be identical. Both results are usable by an expert to identify manageable sets of similar tests for troubleshooting. Additionally, the tree rule would allow correct assessments to assign each group to the right troubleshooting department.

The overall accuracy score of C_2 for both approaches is 90%, which means that C_2 can identify 90% of non-anomalies and anomalies alongside their root cause. With the above simple example, we have defined an automated anomaly detection system that is functional to improve the quality of the networking service. Indeed, TTrees can self-identify network performance issues. So it can be used to alert the right support personnel, either the TCP or radio optimization team, in the presented example, which will receive as input the rules from Fig. 7.10 that raised the alarm. Acknowledging that other classification approaches might yield higher accuracy, we remark that would occur at the expense of interpretability.

7.3.4. TCP Performance Evaluation

TTrees has been executed in three different data-sets to identify anomalies with TCP data exchanges. The target KPI for Dataset #2 is *Average Session Duration,* discretized into six classes, and for Dataset #3 the KPI is *Throughput,* discretized into eight classes.

Table 7.3: Knowledge (C_1) and Aspect Classification (C_2) trees' performance across the different data-sets using multiple aspect selection algorithms (Mutual Information, Joint Mutual Information, and Miscoding)

	Classifier	**Accuracy**	**Precision**	**Recall**	**F1**
Dataset #1 **HTTP** **file DL** **(Nokia)**	C_1	82.5%	NA	NA	NA
	C_2 (MI)	92.0%	NA	NA	NA
	C_2 (JMI)	91.0%	NA	NA	NA
	C_2 (Mscd)	90.0%	NA	NA	NA
Dataset #2 **HTTP** **livepage DL** **(Nokia)**	C_1	82.6%	80.0%	81.5%	80.0
	C_2 (MI)	-	-	-	-
	C_2 (JMI)	-	-	-	-
	C_2 (Mscd)	88.0%	85.0%	86.0%	84.0
Dataset #3 **HTTP** **file DL** **(MONROE)**	C_1	92.0%	90.0%	91.0%	89.0
	C_2 (MI)	95.0%	92.6%	94.4%	92.5
	C_2 (JMI)	95%	94.5%	94.5%	92.7
	C_2 (Mscd)	95.0%	93.0%	94.4%	93.5
Dataset #4 **QUIC** **file DL** **(MONROE)**	C_1	87.3%	NA	NA	NA
	C_2 (MI)	94.0%	NA	NA	NA
	C_2 (JMI)	94.0%	NA	NA	NA
	C_2 (Mscd)	96.0%	NA	NA	NA

Since the new data-sets are large enough for TTrees, we split them into training and test subsets, with 10% and 20% testing data while applying cross validation 30 times, respectively. Note that, due to the small size of Dataset #4, data could not be split into training and testing sets and thus, cross validation could not be carried out, hence the "NA" entries on the table. Tests are done in classifiers C_1 and C_2 to get the metrics shown in Table 7.3. The number of anomalous scenarios is 1152 and 6902, with 25 and 36 most relevant features, respectively. TTrees with Mscd was able to identify similar network aspects in Datasets #2 and #3 as it was in Dataset #1: radio, TCP, and RTT. Unfortunately, the selection of the most relevant features did not work well using MI and JMI for Dataset #2 (HTTP livepage DL), since all the pairs of features selected were redundant, and TTrees could not build C_2 (hence the entries "-" in Table 7.3). This was not surprising to our expert, since the anomalous scenarios of Dataset #2 are very hard to classify, since they contain downloads of webpages with multiple sizes. The fact that Mscd found meaningful network aspects is indeed impressive. For Dataset #1 and #3,

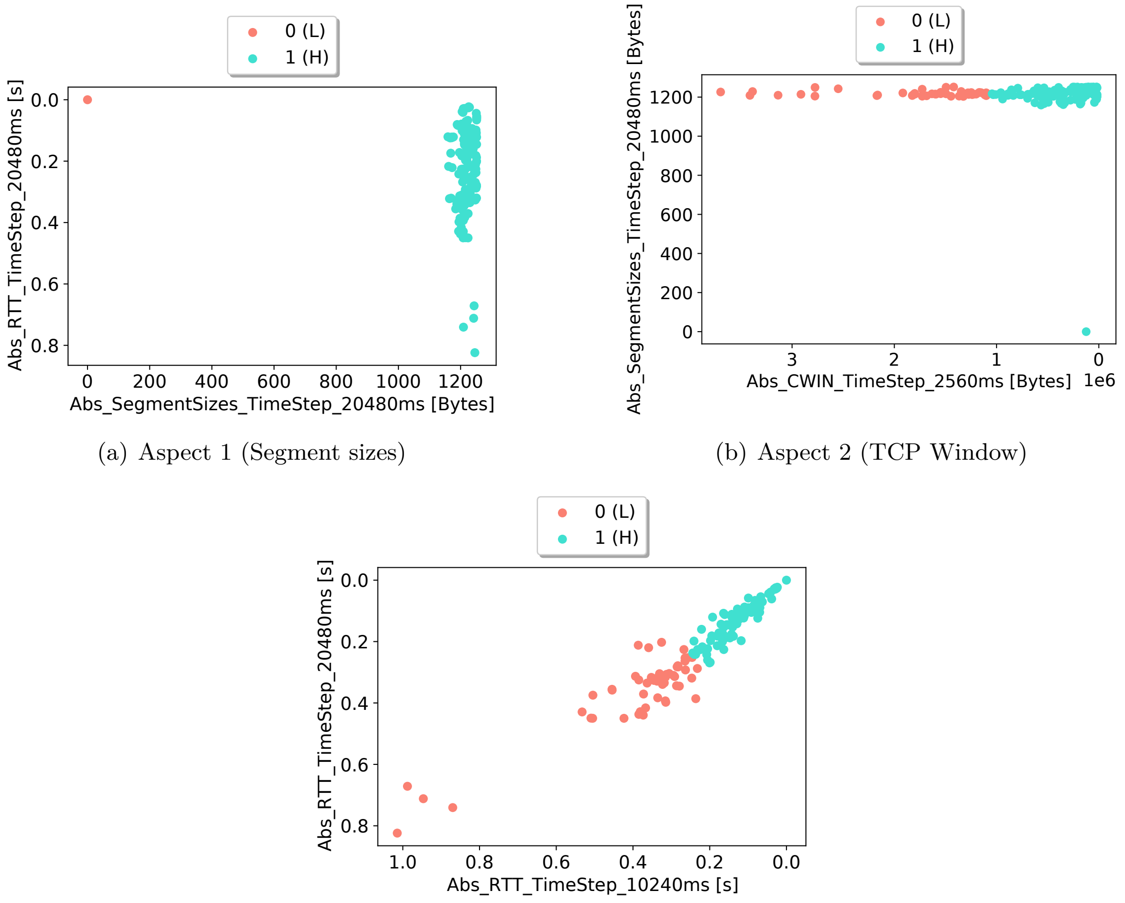
Table 7.4: Classification output of TTrees with QUIC experiments – Data grouped per congestion control algorithm (classes with no samples are omitted)

	Knowledge tree C_1		**Aspect classification tree C_2**	
	Throughput class	**Count**	**Anomaly class**	**Count**
BBR	0	363	111	926
	1	304	110	1
	2	243	011	29
	3	87	010	42
	4	1		
COPA	0	983	111	991
	1	14	011	6
Cubic	0	579	111	907
	1	170	011	9
	2	167	010	35
	3	42	000	7
New Reno	0	562	111	928
	1	212	011	16
	2	164	010	15
	3	40	001	3
			000	36

Table 7.5: Classification output of TTrees with QUIC experiments – Data grouped per experiment execution type (classes with no samples are omitted)

	Knowledge tree C_1		**Aspect classification tree C_2**	
	Throughput class	**Count**	**Anomaly class**	**Count**
Parallel	0	1260	111	1831
	1	413	011	40
	2	208	010	48
	3	70	001	3
			000	29
Sequential	0	1227	111	1921
	1	287	110	1
	2	366	011	20
	3	119	010	44
	4	1	000	14

the C_2 classifiers show similar classification performance (see Table 7.3). However, the use of JMI and MI did not produce network aspects easily identifiable by an expert. In summary, using Mscd leads to better performance than the other two metrics, reaching



(a) Aspect 1 (Segment sizes)

(b) Aspect 2 (TCP Window)

(c) Aspect 3 (RTT)

Figure 7.11: Aspects obtained with TTrees for Dataset #4

good accuracy scores while producing meaningful sets of rules for troubleshooting.

7.3.5. QUIC performance evaluation

Finally, we use TTrees with our fourth data-set, containing the qlogs of almost 4000 experiments carried out with MONROE. The target KPI for Dataset #4 is the average throughput, which has been discretized into five classes. The congestion control algorithm and execution type used in each experiment were recorded jointly with qlogs and MONROE probe context parameters. Table 7.4 summarizes per-congestion control algorithm statistics computed to form the knowledge tree C_1 with the initial steps of the TTrees methodology. The table shows that experiments with COPA, Cubic and New Reno systematically yielded low throughput, especially with COPA, while with BBR we

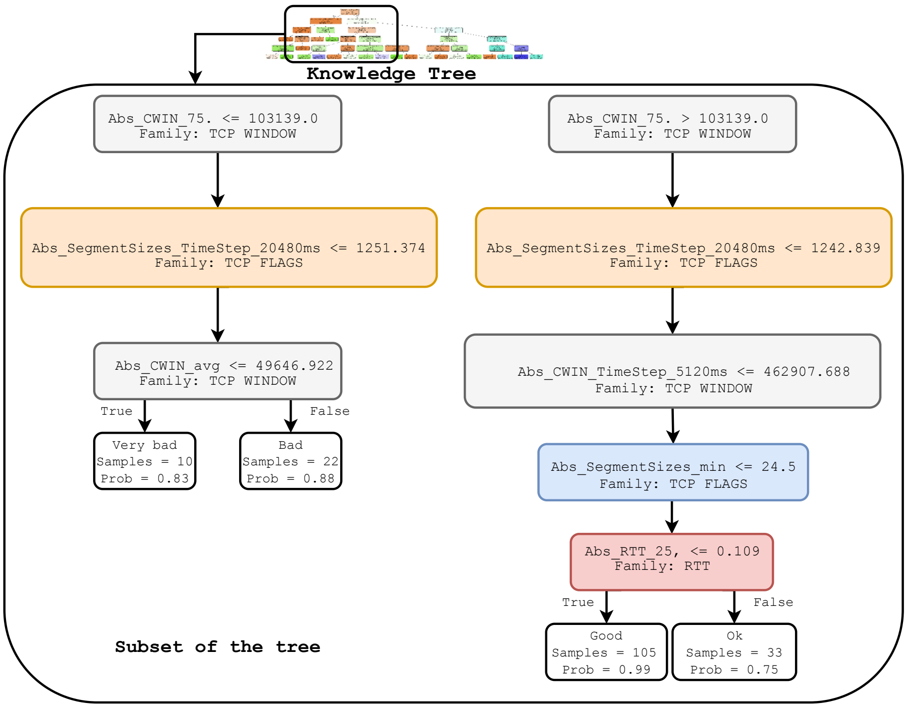


Figure 7.12: Zoom into a subset of the branches of the knowledge tree C_1 built by TTrees for Dataset #4. The figure also reports which family of features the branching variable belongs to (see Table 7.2)

can observe a higher throughput, in terms of distribution over the 5 identified classes. The table also shows the number of anomalous cases classified according to network aspects. There were in total eight possible combinations of three network aspects in which to look for anomalies. The three identified network aspects are shown in Fig.7.11. The first two of them show a clear dependency on one parameter only, which is the segment size in the first case and the congestion window in the second case. Note that small segment sizes would naturally lead to low throughput while small congestion window values are typical of bad performers. So, the selection of those network aspects, which was done automatically by TTrees, makes sense. The third network aspect is a combination of RTT statistics taken with different averaging intervals. Indeed, when it comes to evaluate the throughput of congestion-controlled transmissions, it is well known that RTT statistics are a meaningful network aspect to consider as well. In particular, high RTT values tend to indicate the presence of radio limitations.

Table 7.4 shows that not all combinations of network aspects were observed among

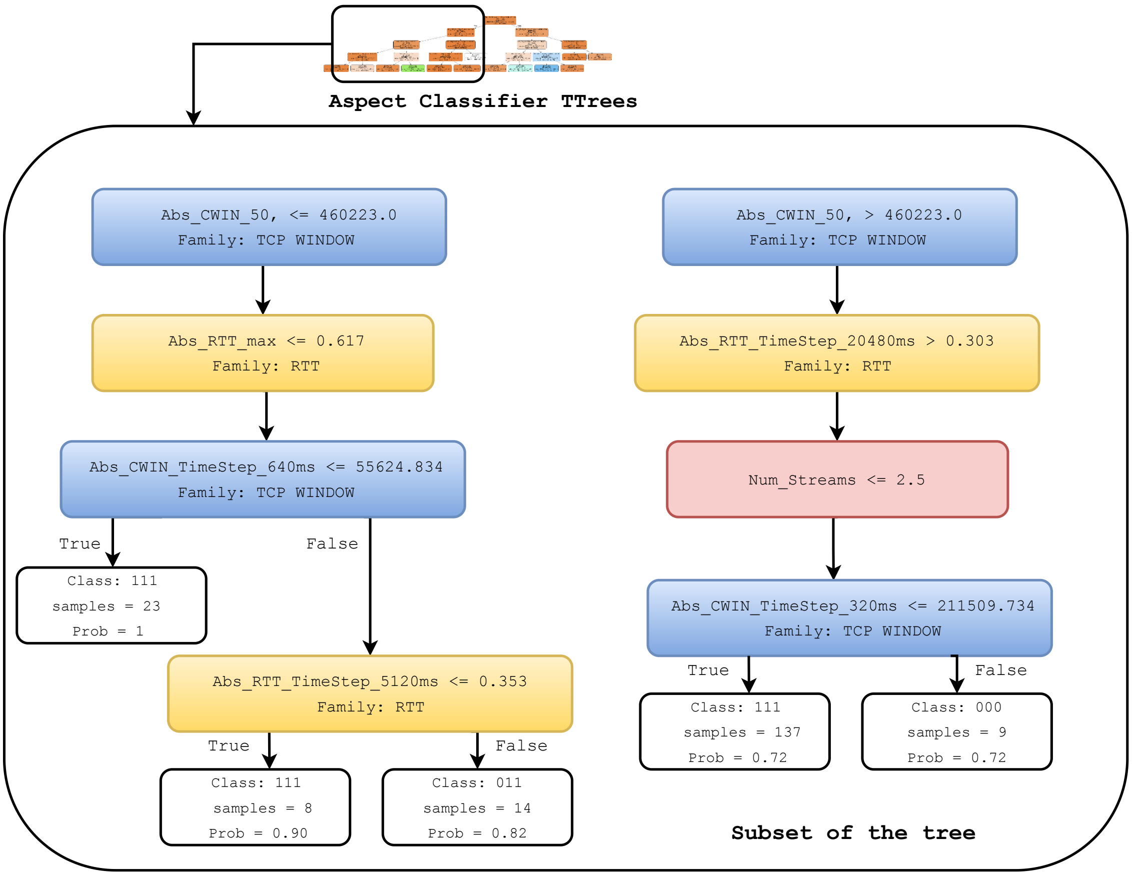


Figure 7.13: Zoom into a subset of the branches of the aspect classification tree C_2 built by TTrees for Dataset #4. The figure also reports which family of features the branching variable belongs to (see Table 7.2)

the identified anomalies. Of course, the more frequently observed case is that there are no anomalies (class 111), while most of detected anomalies show a combination of two or three network aspects. QUIC anomalies with BBR mostly depend on the segment size and RTT, while Cubic and NewReno anomalies also depend on the congestion window. The least number of anomalies was observed with COPA, and are caused by low values of the segment size utilized for transmission. Note that looking at Table 7.4, we can see that no experiment conducted using BBR falls under the second anomaly class, i.e., the congestion window network aspect does not affect BBR anomalies. This is a meaningful result since with BBR the value of the congestion window (CWIN) is automatically modulated to maintain the RTT within acceptable values, so that CWIN cannot be directly correlated with the experienced throughput.

If we consider parallel and serial QUIC data transmissions separately, Table 7.5 resumes the distribution of throughput samples—where of course we can see that sequential downloads observe higher per-download throughput—and anomalies. It is interesting to note that anomalies for parallel download experiments show a fundamentally different distribution across anomaly classes with respect to the case of sequential downloads. For instance, while the first network aspect (the segment size) is identified as a common cause of anomaly in all cases, the second network aspect (i.e., the congestion window size) is more relevant for parallel downloads. This result is meaningful because segment size always affects transmission rate, while the congestion window is affected by the number of ongoing transmission flows.

Note that, due to the small size of Dataset #4, data could not be split into training and testing sets and thus, cross validation could not be carried out. The knowledge tree, shown in Fig. 7.12, yielded 199 anomalous scenarios in total. As mention before, here we selected only three network aspects, which were extracted from a set of nine most relevant features. The aspect selection of TTrees automatically deemed appropriate to stop the search of network aspects to three, because adding a fourth one would have been redundant. We remark that this was done automatically, and we did not have to manually change the number of network aspects to select. The accuracy of the aspect classification tree for the QUIC data-set of Fig. 7.13, was the highest of all C_2 trees, with a value of 96% and includes the three types of anomalies identified. Also for the case of QUIC, using miscoding rather than (join) mutual information for aspect selection was eventually advantageous in terms of accuracy, as visible in Table 7.3.

With Dataset #4, trees C_1 and C_2 are smaller than with the other data-sets, which allows us to analyze them more in detail. A general analysis of the classification trees C_1 (partially depicted in Fig. 7.12) and C_2 (Fig. 7.13) shows that critical QUIC features were properly selected to classify the performance of tests. In particular congestion window size (specifically, its 75th percentile) is at the root node of C_1. Indeed, given the characteristics of the tests performed using different congestion algorithms and assuming similar radio conditions, it is clear that the main features differentiating each type of test has to be the congestion window related statistics. Other features like the number of streams, buffering/delay (RTT), packet loss or segment sizes are also correctly classifying each test. For example, RTT values above 100 ms identify lower throughput samples due to capacity limitations (high buffering). A higher number of streams is also normally identifying better samples in the knowledge tree C_1 built by TTrees for Dataset #4.

The aspect classification tree C_2 tells that, with high probability, we either have an anomaly due to a low congestion window (maybe due to packet loss), or a combination of the three identified network aspects. For the first case we have two leaves of the tree, and in both leaves we can see that CWIN and RTT in the initial five seconds were OK, but at the end of the test they were not. This might be due to packet loss. In the case of combined anomalies, the tree shows that there was low radio capacity or the congestion algorithm was too aggressive. In both cases we have identified the cause of anomalous performance of QUIC with the same methodology used to identify anomalous performance of TCP.

7.3.6. Modifications needed to deal with imbalancedness of data-sets

To further extend the evaluation of TTrees, we employ Datasets #5 and #6, collected with MONROE. These data-sets are collected from real operational mobile networks over which we do not have control. However, we have tested the connectivity to Facebook and Google webpages, which are highly replicated across the network, and easily reachable without having to traverse large portions of the cellular and core networks of the mobile operators to which MONROE probes were connected. As such, these data-sets suffer imbalancedness, meaning that TCP problems are rare, while radio conditions might impair performance with much higher frequency, especially for mobile MONROE probes. Therefore, the analysis of Datasets #5 and #6 is suitable to showcase possible drawbacks due to the use of simple ML mechanisms in TTrees, and in particular the clustering operated with k-means, which is known to suffer in case of parsing imbalanced data-sets [112, 113].

Table 7.6 summarizes the results obtained with TTrees in terms of accuracy and the

Table 7.6: A summary of the knowledge tree's (C_1) performance in TTrees utilizing Facebook & Google as a service in all the countries part of the MONROE project with all their operators.

Services	Country	Operators	Accuracy (%)	# Samples	# Anomalous samples
Facebook (Dataset #5)	**Sweden**	*op*0_sw	96.0	183240	3162
		*op*2_sw	96.6	235568	3346
	Norway	*op*0_no	96.5	160290	2235
		*op*1_no	96.6	130334	2139
		*op*2_no	95.4	128804	2688
	Italy	*op*0_it	96.9	66420	1008
		*op*1_it	96.5	83966	1049
	Spain	*op*0_es	97.5	26795	320
		*op*1_es	95.5	30575	380
		*op*2_es	96.3	86450	1044
Google (Dataset #6)	**Sweden**	*op*0_sw	92.7	146327	3061
		*op*2_sw	98.9	180784	953
	Norway	*op*1_no	99.2	127628	469
	Italy	*op*0_it	96.8	66420	1008
		*op*1_it	96.5	83966	1049
	Spain	*op*0_es	98.4	24530	117
		*op*1_es	91.4	2078	32718
		*op*2_es	88.5	77729	7145

number of samples concerning the knowledge tree for Datasets #5 and #6. The table shows results classified according to the origin of the data collected, i.e., per country of test and per operator. Similarly, Table 7.7 showcases the performance of TTrees when it comes to the aspect classification tree for the same data-sets. For both data-sets, the accuracy obtained with TTrees in both trees C_1 and C_2 is high, and the number of anomalies is as limited as it could be expected. However, Table 7.8 reveals that TTrees does not always manage to reach a fully meaningful classification of anomaly causes. In particular, the fourth and fifth columns of Table 7.8 show that the number of aspects identified with TTrees, and of the corresponding families, by means of k-means clustering is small for these data-sets. This is a problem because the less aspects and families are extracted by the algorithm, the poorer is the precision with which the cause of an anomaly can be explained. Moreover, Table 7.8 does not report the results for some of the tested networks because in those cases there were not enough aspects to generate C_2. The problem with the analysis of these data-sets is that the anomalies found by means of

Table 7.7: A summary of the aspect classification tree's (C_2) performance with TTrees utilizing Facebook & Google as a service in all the countries part of the MONROE project with all their operators.

Services	**Country**	**Operators**	**Accuracy (%)**	**# Samples**	**# Anomalous samples**
Facebook (Dataset #5)	**Sweden**	*op*0_sw	98.5	183240	104
		*op*2_sw	98.9	235568	295
	Norway	*op*0_no	98.9	160290	2235
		*op*1_no	98.6	130334	2139
		*op*2_no	98.8	128804	770
	Italy	*op*1_it	98.7	83966	5
	Spain	*op*0_es	99.0	26795	18
		*op*1_es	98.9	30575	38
		*op*2_es	99.0	86450	1044
Google (Dataset #6)	**Sweden**	*op*0_sw	98.5	146327	105
		*op*2_sw	98.9	180784	322
	Norway	*op*1_no	99.6	127628	23
	Italy	*op*0_it	98.9	66420	89
		*op*1_it	99.0	83966	0
	Spain	*op*0_es	99.5	24530	0
		*op*1_es	94.5	24530	32718
		*op*2_es	95.5	77729	350

C_1 are strongly imbalanced. For instance, Figure 7.15 shows that k-means founds most of the anomalies of C_1 concentrated in one of the two clusters extracted after applying binary classification using the most relevant features. In some scenarios, such as the ones displayed in those figures, this led to the impossibility of finding a set of features that fell under the balancedness threshold, and, thus, no aspects were obtained for labelling anomalous classes in C_2. Therefore, applying k-means on these imbalanced C_1 anomalies can incur performance problems, as actually witnessed in our experiments.

Note that we have designed our CIAN methodology in a modular way, and the TTrees implementation relaying on k-means is only a possible implementation with some strong advantages (it relies on simple algorithms and is interpretable) and some cons due to the limitations of the selected algorithms. However, in CIAN, methodological phases are well separated and the overall system is highly modular. Therefore, tools and algorithm used for a particular implementation can be easily exchanged guaranteeing no inter-dependency between technologies at different steps. Considering this, we can think of combating imbalancedness through a relatively simple modification of TTrees to implement CIAN.

The alternative implementation of the methodology can simply differ from TTrees in how problems are clustered. In particular, we tested a modified version of TTrees obtained by interchanging k-means with Gaussian Mixture Model (GMM) [114], which is robust to imbalancedness and operates clustering in a very flexible way, although with less interpretable results.

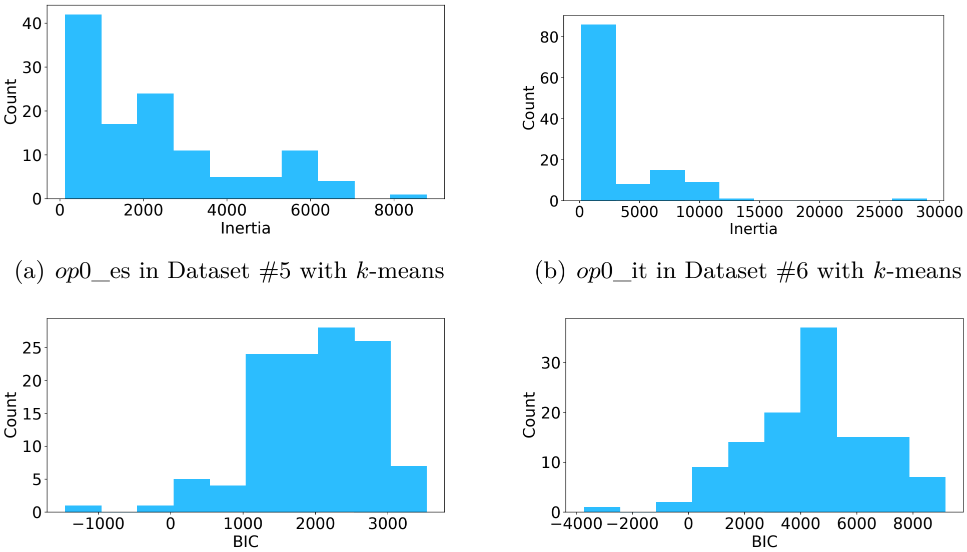
7.3.6.1. Modified TTrees with GMM clustering

It is important to bear in mind that GMM is a model-based clustering approach that works with the concept of soft clustering, meaning that data is not just assigned to a cluster, but instead it has a likelihood of belonging to it [114]. Therefore, the objective of applying GMM to implement CIAN is to maximize this probability, a concept which can be directly associated to that of cluster tightness measured through inertia. Indeed, the previously defined TTrees implementation computes the inertia score for each pair of attributes. Instead, GMM requires to use of a different metric to draw performance comparisons between different pairs of attributes. The most well-known score for the selection of the optimal model in model-based clustering is the Bayesian Information Criterion (BIC) [115], so we use the BIC metric. Like inertia, the BIC metric is a penalty measure, which has to be minimized.

To show that GMM and k-means behave in a substantially different way, we evaluate the behavior of BIC and inertia obtained by applying GMM and k-means to form C_2 with the data relative to two operators within what collected in Dataset #5. Specifically, Figure 7.14 depicts the clustering performance values for the pairs of attributes extracted for Dataset #5 with $op0$ in Spain, using GMM or k-means, respectively. Given that having lower inertia and BIC values translates into better separation between clusters, we see that the BIC metric reports values that are clearly lower than the ones extracted with inertia, and, more importantly, grouped within a considerably smaller range. This difference in the consistency of clustering performance may be an indicative of unstable behaviour of k-means, thus showing the presence of imbalancedness in the data-set. Moreover, according to the analysis conducted, low inertia would correspond to values between a 40 and 100, which as it is shown in the image, it is not the case for most attributes. This translates in the obtained clusters not being the most optimal, highlighting an under-performance of k-means compared to previous data-sets and resulting of this imbalancedness. Figure 7.14 shows that the same analysis carried out for another operator and data-set (in Italy and with Dataset #6) yields an upper threshold for inertia values higher than the one obtained with BIC and a similar tendency in the distribution of data compared with the scenario mentioned previously. This proves that the imbalancedness of the data-set is consistent and observable across multiple operators and scenarios and thus it is a direct result of the nature of the experiments conducted.

Thus, using BIC, GMM is able to identify substantially different behaviors across the data collected for one operator or the other, while k-means might not be able to do so, at least when the data-sets are imbalanced. Figure 7.15 clearly shows that the anomalies obtained from C_1 are indeed imbalanced for the two cases analyzed in Figure 7.14. During the clustering phase of the methodology, samples classified as anomalous are fit into one of two clusters for each aspect: a cluster of problematic samples and one of compliant (non-problematic) samples. Since “N Class 0”, which is the ratio of samples falling under the non-problematic class for this binary classification problem, only takes values that are very close to 0 or 1, the anomalies are indeed imbalanced, as opposed to being evenly distributed.

By comparing aspect classification performance across more scenarios, the sixth and seventh columns of Table 7.8 show that using GMM instead of k-means, with Datasets #5 and #6 finds more aspects. The last two column show that GMM finds most of the aspects found by k-means and identifies additional ones. This shows that GMM tends to be more general than k-means. However, the results obtained with k-means are more interpretable. Interpretability in decision trees is tightly correlated to the tree’s size. Thus, the higher the number of nodes, the less interpretable the tree is. The significant difference in the number of aspects identified with GMM translates in an increase in the depth and branching of C_2. Therefore, although GMM was able to compute a solution in those particular cases, it yielded clusters and an aspect classification trees harder to



(a) *op*0_es in Dataset #5 with *k*-means

(b) *op*0_it in Dataset #6 with *k*-means

(c) *op*0_es in Dataset #5 with GMM

(d) *op*0_it in Dataset #6 with GMM

Figure 7.14: Histogram of Inertia (a, b) and BIC (c, d) values extracted during aspect clustering for all relevant aspects and with two operators in Spain and Italy. The lower the BIC or inertia the cleaner cluster grouping and splitting of samples is overall

traverse and understand. This relates to the fact that although GMM can handle complex patterns in data, it does so at the expense of interpretability [116].

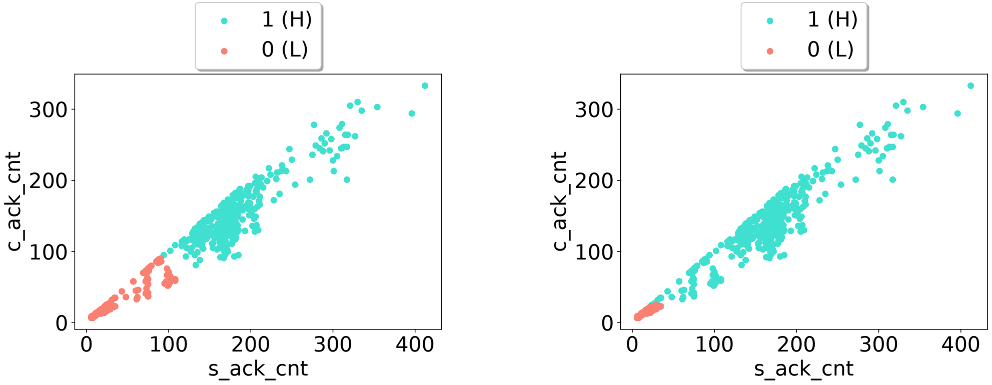
7.3.6.2. Take away message

In case the data-set is balanced *k*-means and GMM convey similar clustering each grouping the clusters based on how the algorithm works underneath as shown in Figures 7.16 and 7.17. This is true for Datasets #1 to #4. However, Table 7.8 tells that the differences in the aspect families extracted from the application of *k*-means and GMM are significant in Datasets #5 and #6. The table reports the number of intersecting aspects and the ratio between intersecting aspects and the size of the smallest aspect set among the ones identified by *k*-means and GMM. The value of the ratio is, in most cases, close to one, which hints to the fact that the intersection of aspects found with the two approaches is similar to the smallest set identified by either GMM or *k*-means. Moreover, it is significant that in all the subsets available in Datasets #5 and #6, GMM finds a group of attributes equal or larger to the one extracted with *k*-means. In practice, as shown in the table, all the aspect families detected by *k*-means are extended by GMM, which is indicative of the fact that k-means identifies a subset of the data extracted with GMM. Thus, differently from GMM, k-means enforces a more aggressive filtering behaviour when selecting aspects, and this behavior is strongly affected by the balancedness filter used in the proposed methodology.

This difference between using GMM or k-means is clearly displayed, e.g., in Figure 7.15, in which we can see how GMM is able to identify a significantly higher number of aspects closer to an ideally balanced distribution of samples. It is also important to note that in some cases, k-means does not find a sufficient number of aspects and is not capable of identifying the root for the existence of anomalies. In fact, when TTrees with k-means does not obtain an aspect label, no aspect families are highlighted in C_2, and TTrees cannot identify clearly whether a group of anomalies as classified by the decision tree C_1 is problematic or not for the target KPI. Indeed, the fact that some operators are not included in Table 7.8 alludes to the failure of some experiments utilizing k-means. We observed that TTrees with the latter clustering algorithm was not able to generate C_2 due to a highly imbalanced number of anomalies obtained from C_1. On the contrary, the application of GMM as the clustering algorithm in the same scenarios always resulted in a complete aspect classification tree. Note also that, although GMM was able to compute a solution in those particular cases, it yielded clusters and an aspect classification trees harder to traverse and understand. Therefore, at the end of the day, if imbalanced anomalies are encountered, a feasible approach would be to switch the clustering algorithm to one that is able to handle this particular data distribution, thus, allowing for a higher flexibility.

Table 7.8: A comparison of the number of aspects and aspect families extracted from C_2 between k-means and GMM for the Google & Facebook data-sets in all the countries part of the MONROE project. The table also shows the number of aspects that are common to both algorithms as well as the ratio, calculated from the division of the intersection by the size of the smallest set of aspects between k-means and GMM.

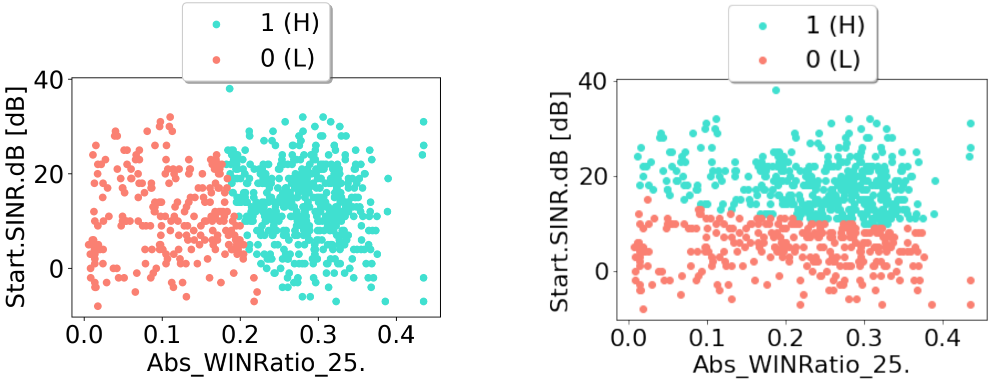
Service	**Country**	**Operators**	**# k-means Aspects**	**k-means Aspect families**	**# GMM Aspects**	**GMM Aspect families**	**# Aspect intersection**	**Ratio**
Facebook (Dataset #5)	**Sweden**	*op*0_sw	6	RTT, TCP Window, Packet anomaly	8	RTT, TCP, TCP Window, Packet anomaly	4	0.667
		*op*2_sw	4	RTT, Packet anomaly	10	RTT, TCP, TCP Window, Packet anomaly	4	1.0
	Norway	*op*0_no	2	-	11	RTT, TCP, TCP Window, Packet anomaly	2	1.0
		*op*1_no	2	-	10	RTT, TCP, TCP Window, Packet anomaly	1	0.5
		*op*2_no	5	RTT, TCP, TCP Window	10	RTT, TCP Window, Packet anomaly	4	0.8
	Italy	*op*1_it	8	RTT, TCP, TCP Window, Packet anomaly	10	RTT, TCP, TCP Window, Packet anomaly	4	0.5
	Spain	*op*0_es	4	RTT, TCP	7	RTT, TCP, TCP Window	3	0.75
		*op*1_es	6	RTT, TCP	7	RTT, TCP, TCP Window	5	0.833
		*op*2_es	2	-	7	TCP, TCP Window, Packet Anomaly	2	1.0
Google (Dataset #6)	**Sweden**	*op*0_sw	10	RTT, TCP, TCP Window	10	RTT, TCP, TCP Window, Packet anomaly	4	0.4
		*op*2_sw	8	RTT, TCP, TCP Window	9	RTT, TCP, TCP Window	7	0.875
	Norway	*op*1_no	5	RTT, TCP Window, Packet anomaly	9	RTT, TCP Window, Packet anomaly	5	1.0
	Italy	*op*0_it	9	RTT, TCP, TCP Window	8	RTT, TCP, TCP Window	7	0.875
		*op*1_it	10	RTT, TCP, Packet Anomaly	12	RTT, TCP, Packet Anomaly	8	0.8
	Spain	*op*0_es	1	-	1	-	1	1.0



(a) Aspect 1 (Packet anomalies) with *k*-means

(b) Aspect 1 (Packet anomalies) with GMM

Figure 7.16: Cluster grouping using *k*-means and GMM as the cluster algorithm using Google Dataset and *op*0_*it*

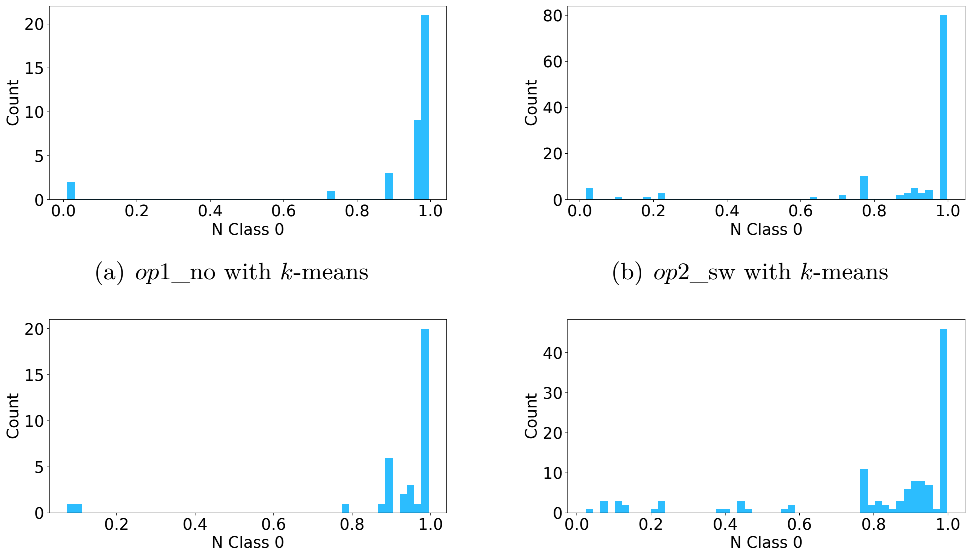


(a) Aspect 1 (TCP Congestion Window Vs Signal-To-Noise-plus-Interference Ratio (SINR)) with *k*-means

(b) Aspect 1 (TCP Congestion Window Vs Signal-To-Noise-plus-Interference Ratio (SINR)) with GMM

Figure 7.17: Cluster grouping Comparsion using *k*-means and GMM as the cluster algorithm for Dataset #1

Ideally, we want an easy-to-interpret separation to distinguish between anomaly samples, and this is why in TTrees we have opted for *k*-means. However, in case of imbalanced data-sets, a more complex separation of anomalies into clusters is needed, and can be achieved by using GMM, at the expenses of interpretability. Therefore, at the end of the day, if anomalies are balanced, *k*-means is a suitable tool to be used to implement



(a) *op*1_no with *k*-means

(b) *op*2_sw with *k*-means

(c) *op*1_no with GMM

(d) *op*2_sw with GMM

Figure 7.15: Histogram of the ratio of samples falling under the non-problematic class during aspect clustering for all relevant aspects in Dataset #5 and with two operators in Sweden and Norway. As it can be seen, most aspects, especially with *k*-means, yield imbalanced clusters of anomalies.

CIAN, whereas, should imbalanced anomalies be encountered, a feasible approach would be to switch to GMM. In practice, we could extend TTrees by automatically checking the presence of imbalanced anomalies, and then automatically select the appropriate clustering algorithm between *k*-means and GMM, or another one that is able to handle the particular data/anomaly distribution observed in the data-set. This would allow for a higher flexibility in the implementation of CIAN.

7.4. Discussion

In this Chapter, we have developed, validated and applied a methodology to identify the root causes of network anomalies by means of interpretable ML algorithms. Our methodology, named TTrees, allows for automatic identification of the networking aspects that cause anomalous behaviors. Such anomalies cannot be immediately and directly explained by observing the network features. In fact, the potentially large number of network aspects, even in the presence of a limited number of features and samples per feature, makes it impossible to manually inspect and correlate. With TTrees, we have purposely developed a radically novel methodology and implemented it with Python. Specifically, we have leveraged the key observation that if a network behavior cannot be modeled (or learned by ML), it is a symptom of network anomaly. We therefore easily identify performance anomalies, after which we are able to use unsupervised ML and Kolmogorov complexity-inspired tools to smartly search for the aspects that are more relevant to explain where the anomaly is rooted. In particular we have introduced a novel metric, miscoding, for the evaluation of redundancy and relevance of features. The final outcome of our proposed methodology is a set of easy-to-interpret classification rules, that, in case of performance anomaly, allow for automatically alerting the appropriate departments for corrective actions. To do so, TTrees only needs little volumes of samples for the features. TTrees is not specifically designed for a network protocol or service or data-set type, as we have validated in this work by applying TTrees to the analysis of TCP and QUIC with data gathered with different granularity, formats and richness of experimentally collected metrics. Indeed, TTrees allows to fully automatize network troubleshooting with high accuracy and fast training, as shown in this Chapter with the help of real data gathered from operational cellular networks in various countries.

8 Conclusions

This Thesis investigated WebRTC and QUIC in different MBB networking scenarios while bringing meaningful insights into the behavior of these protocols in real operational cellular networks with the help of the MONROE platform. We have presented different ideas on assessing the overall QoE and QoS using stats collected through qlogs and WebRTC internals and also used data collected from the MONROE testbed to further build methodologies that automate the detection of network problems and help fast cellular network troubleshooting leveraging ML algorithms.

As such, in Part II, we have presented a complete and novel methodology for evaluating Web services using operational MBB networks. Initially, we have observed that mobility poses a challenge for WebRTC, considering MBB operators do not yet provide full quality coverage for users on the move.

We continued investigating the overall performance of rising protocols, namely QUIC and HTTP3, under different network conditions various congestion control algorithms, in Chapter 5. Containers for WebRTC and QUIC experiments, as well as all the raw measurements, have been made available online.

In Part III of the Thesis, we have demonstrated the usefulness of applying data science to data collected from operational commercial cellular networks. To elaborate further, Chapter 6 has proposed a novel supervised methodology for the detection of network anomalies using interpretable ML algorithms. The work reported in this Thesis has shown that the main advantages of the proposed STrees methodology can be fully automated, and its results are easy to understand. Concretely, the methodology is based on the combined application of well-established supervised and unsupervised ML tools. We have also presented a few application examples based on real data from operational cellular networks.

Finally, Chapter 7 has presented a generalization of the STrees methodology with fully automation and unsupervised algorithms. Indeed, by leveraging unsupervised ML and Kolmogorov complexity-inspired tools to smartly search for the aspects that are more relevant to explain where the anomaly is rooted in the network protocols, we have proposed CIAN. In CIAN, we have also introduced a novel metric, miscoding, to evaluate the redundancy and relevance of performance indicators. In summary, the outcome of our proposed STrees and CIAN methodologies are a set of easy-to-interpret classification rules that automatically allow us to alert the appropriate departments for corrective actions in case of performance anomaly. Combining performance evaluation over rising protocols and collected data from real opertional MBB while building methdologies to automate the fault detection in cellular networks which are easy to interpret reflect the key noviltles behind this Thesis.

References

[1] M. Moulay and V. Mancuso, “Experimental performance evaluation of webrtc video services over mobile networks,” in *IEEE INFOCOM 2018 _ IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 541–546.

[2] C. Midoglu, M. Moulay, V. Mancuso, O. Alay, A. Lutu, and C. Griwodz, “Open video datasets over operational mobile networks with monroe,” in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys ’18. New York, NY, USA: Association for Computing Machinery, 2018, p. 426–431. [Online]. Available: https://doi.org/10.1145/3204949.3208138

[3] M. Moulay, F. D. Munoz, and V. Mancuso, “On the experimental assessment of QUIC and congestion control schemes in cellular networks,” in *2021 19th Mediterranean Communication and Computer Networking Conference (MedComNet) (MedComNet 2021)*, Online Conference, Jun. 2021.

[4] M. Moulay, R. García, P. J. R. Maroni, J. Lazaro, V. Mancuso, and A. Fernández Anta, “A novel methodology for the automated detection and classification of networking anomalies,” in *IEEE INFOCOM 2020 _ IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2020, pp. 780–786.

[5] M. Moulay, R. García, V. Mancuso, P. Rojo, and A. Fernández Anta, “TTrees: automated classification of causes of network anomalies with little data,” in *2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM) (WoWMoM 2021)*, Pisa, Italy, Jun. 2021.

[6] V. Mancuso, M. Peón Quirós, C. Midoglu, M. Moulay, V. Comite,

A. Lutu, Özgü Alay, S. Alfredsson, M. Rajiullah, A. Brunström, M. Mellia, A. Safari Khatouni, and T. Hirsch, "Results from running an experiment as a service platform for mobile broadband networks in europe," *Computer Communications*, vol. 133, pp. 89–101, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366417312860

[7] M. Moulay, R. García, V. Mancuso, P. Rojo, and A. Fernández Anta and Ali Safari Khatouni, "Montrees: Automated detection and classification of networking anomalies in cellular networks," *IEEE Transactions on Network and Service Management*, pp. 1–1, 2021.

[8] M. Moulay, R. García, P. Rojo, F. Diez, V. Mancuso, and A. Fernández Anta, "Automated identification of network anomalies and their causes with interpretable machine learning: the ttrees methodology," *Computer Communications*, pp. 1–1, 2021.

[9] T. Ogata, A. Takeuchi, S. Fukuda, T. Yamada, T. Ochi, K. Inoue, and J. Ota, "Characteristics of skilled and unskilled system engineers in troubleshooting for network systems," *IEEE Access*, vol. 8, pp. 80 779–80 791, 2020.

[10] A. Zeidan, A. Lehmann, and U. Trick, "WebRTC enabled multimedia," in *in proceedings of World Telecommunications Congress 2014*, Jun. 2014.

[11] A. Johnston, J. Yoakum, and K. Singh, "Taking on WebRTC in an enterprise," *IEEE Communications Magazine*, vol. 51, no. 4, pp. 48–54, April 2013.

[12] S. Loreto and S. P. Romano, "How Far Are We from WebRTC_1.0? An Update on Standards and a Look at What's Next," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 200–207, 2017.

[13] R. Marx, J. Herbots, W. Lamotte, and P. Quax, "Same standards, different decisions: A study of QUIC and HTTP/3 implementation diversity," in *Proceedings of ACM EPIQ*, 2020.

[14] B. Garcia, F. Gortazar, L. Lopez_Fernandez, M. Gallego, and M. Paris, "WebRTC Testing: Challenges and Practical Solutions," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 36–42, 2017.

[15] B. Grozev, G. Politis, E. Ivov, T. Noel, and V. Singh, "Experimental Evaluation of Simulcast for WebRTC," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 52–59, 2017.

[16] P. K. Kharat, A. Rege, A. Goel, and M. Kulkarni, "QUIC protocol performance in wireless networks," in *Proceedinfs of ICCSP*, 2018.

[17] P. Wang, C. Bianco, J. Riihijärvi, and M. Petrova, "Implementation and performance evaluation of the QUIC protocol in Linux kernel," in *Proceedings of ACM MSWIM*, 2018.

[18] J. P. Santos, R. Alheiro, L. Andrade, A. L. Valdivieso_Caraguay, L. I. Barona_López, M. A. Sotelo_Monge, L. J. Garcia_Villalba, W. Jiang, H. Schotten, J. M. Alcaraz_Calero, Q. Wang, and M. J. Barros, "SELFNET framework self_healing capabilities for 5G mobile networks," *Trans. Emerg. Telecommun. Technol.*, vol. 27, no. 9, p. 1225–1232, Sep. 2016. [Online]. Available: https://doi.org/10.1002/ett.3049

[19] A. Asghar, H. Farooq, and A. Imran, "Self_healing in emerging cellular networks: Review, challenges, and research directions," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1682–1709, 2018.

[20] O. Loyola_González, "Black_box vs. white_box: Understanding their advantages and weaknesses from a practical point of view," *IEEE Access*, vol. 7, pp. 154 096–154 113, 2019.

[21] A. HoLzinger, M. Plass, K. HoLzinger, G. Crisan, C. Pintea, and V. Palade, "A glass_box interactive machine learning approach for solving np_hard problems with the human_in_the_loop (2017)," *arXiv preprint arXiv:1708.01104*, 2017.

[22] F. K. Došilović, M. Brčić, and N. Hlupić, “Explainable artificial intelligence: A survey,” in *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 2018, pp. 0210–0215.

[23] R. Marx, W. Lamotte, J. Reynders, K. Pittevils, and P. Quax, “Towards QUIC debuggability,” in *Proceedings of ACM EPIQ*, 2018.

[24] FCC, “2013 Measuring Broadband America February Report,” FCC’s Office of Engineering and Technology and Consumer and Governmental Affairs Bureau, Tech. Rep., 2013.

[25] E. Halepovic, J. Pang, and O. Spatscheck, “Can you GET me now?: Estimating the time_to_first_byte of HTTP transactions with passive measurements.” in *Proc. of IMC*, 2012.

[26] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, “A First Look at Cellular Network Performance during Crowded Events,” in *Proc. of SIGMETRICS*, 2013.

[27] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. Mao, S. Sen, and O. Spatscheck, “An In_depth Study of LTE: Effect of Network Protocol and Application Behavior on Performance,” in *Proc. of SIGCOMM*, 2013.

[28] Tektronix, “Reduce Drive Test Costs and Increase Effectiveness of 3G Network Optimization,” Tektronix Comm., Tech. Rep., 2009.

[29] C. Kreibich, N. Weaver, B. Nechaev, and V. Paxson, “Netalyzr: Illuminating the edge network,” in *Proc. of the 10th ACM SIGCOMM conference on Internet measurement*, 2010, pp. 246–259.

[30] N. Vallina_Rodriguez, “Illuminating the Third Party Mobile Ecosystem with the Lumen Privacy Monitor,” in *FTC PrivacyCon 2017*, January 2017.

[31] M. R. Fida, A. Lutu, M. K. Marina, and O. Alay, “Zipweave: Towards efficient

and reliable measurement based mobile coverage maps," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.

[32] S. Taheri, L. A. Beni, A. V. Veidenbaum, A. Nicolau, R. Cammarota, J. Qiu, Q. Lu, and M. R. Haghighat, "Webrtcbench: a benchmark for performance assessment of webrtc implementations," in *2015 13th IEEE Symposium on Embedded Systems For Real-time Multimedia (ESTIMedia)*, 2015, pp. 1–7.

[33] B. Sredojev, D. Samardzija, and D. Posarac, "Webrtc technology overview and signaling solution design and implementation," *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pp. 1006–1009, 2015.

[34] V. Singh, A. Abello Lozano, and J. Ott, "Performance analysis of receive-side real-time congestion control for webrtc," in *2013 20th International Packet Video Workshop*, 2013, pp. 1–8.

[35] C. C. Spoiala, A. Calinciuc, C. O. Turcu, and C. Filote, "Performance comparison of a webrtc server on docker versus virtual machine," in *2016 International Conference on Development and Application Systems (DAS)*, 2016, pp. 295–298.

[36] P. Biswal and O. Gnawali, "Does QUIC make the Web faster?" in *Proceedings of IEEE GLOBECOM*, 2016.

[37] G. Carlucci, L. De Cicco, and S. Mascolo, "HTTP over UDP: An experimental investigation of QUIC," in *Proceedings of ACM SAC*, 2015.

[38] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita_Rotaru, and A. Mislove, "Taking a long look at QUIC: An approach for rigorous evaluation of rapidly evolving transport protocols," in *Proceedings of ACM IMC*, 2017.

[39] A. Langley, A. Riddoch, A. Wilk, A. Vicente, C. Krasic, D. Zhang, F. Yang, F. Kouranov, I. Swett, J. Iyengar, J. Bailey, J. Dorfman, J. Roskind, J. Kulik, P. Westin, R. Tenneti, R. Shade, R. Hamilton, V. Vasiliev, W. Chang, and

Z. Shi, “The QUIC transport protocol: Design and internet_scale deployment,” in *Proceedings ACM SIGCOMM*, 2017.

[40] M. Nguyen, H. Amirpour, C. Timmerer, and H. Hellwagner, “Scalable high efficiency video coding based HTTP adaptive streaming over QUIC,” in *Proceedings of ACM EPIQ*, 2020.

[41] J. Rüth, K. Wolsing, K. Wehrle, and O. Hohlfeld, “Perceiving QUIC: Do users notice or even care?” in *Proceedings of ACM CoNEXT*, 2019.

[42] S. Cook, B. Mathieu, P. Truong, and I. Hamchaoui, “QUIC: Better for what and for whom?” in *Proceedings of IEEE ICC*, 2017.

[43] T. A. B. Alexander Yu, “Dissecting performance of production QUIC,” in *Proceedings of ACM EPIQ*, 2021.

[44] T. Marwala, *Causality, Correlation and Artificial Intelligence for Rational Decision Making*. WORLD SCIENTIFIC, 2015. [Online]. Available: https://www.worldscientific.com/doi/abs/10.1142/9356

[45] B. Huang, K. Zhang, J. Zhang, J. Ramsey, R. Sanchez-Romero, C. Glymour, and B. Schölkopf, “Causal discovery from heterogeneous/nonstationary data with independent changes,” 2019. [Online]. Available: https://arxiv.org/abs/1903.01672

[46] A. McGovern, R. Lagerquist, D. J. Gagne, G. E. Jergensen, K. L. Elmore, C. R. Homeyer, and T. Smith, “Making the black box more transparent: Understanding the physical implications of machine learning,” *Bulletin of the American Meteorological Society*, vol. 100, no. 11, pp. 2175 – 2199, 2019. [Online]. Available: https://journals.ametsoc.org/view/journals/bams/100/11/bams-d-18-0195.1.xml

[47] J. Pearl, “The seven tools of causal inference, with reflections on machine learning,” *Commun. ACM*, vol. 62, no. 3, p. 54–60, feb 2019. [Online]. Available: https://doi.org/10.1145/3241036

[48] D. Garreau and U. von Luxburg, “Explaining the explainer: A first theoretical analysis of lime,” in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, ser. Proceedings of Machine Learning Research, vol. 108. PMLR, Aug. 2020, pp. 1287–1296. [Online]. Available: http://proceedings.mlr.press/v108/garreau20a.html

[49] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions,” in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS’17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 4768–4777.

[50] A. Shrikumar, P. Greenside, and A. Kundaje, “Learning important features through propagating activation differences,” in *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ser. ICML’17. JMLR.org, 2017, p. 3145–3153.

[51] A. Binder, G. Montavon, S. Bach, K. Müller, and W. Samek, “Layer-wise relevance propagation for neural networks with local renormalization layers,” *CoRR*, vol. abs/1604.00825, 2016. [Online]. Available: http://arxiv.org/abs/1604.00825

[52] R. Barco, L. Nielsen, R. Guerrero, G. Hylander, and S. Patel, “Automated troubleshooting of a mobile communication network using bayesian networks,” in *4th International Workshop on Mobile and Wireless Communications Network*, 2002, pp. 606–610.

[53] R. M. Khanafer, B. Solana, J. Triola, R. Barco, L. Moltsen, Z. Altman, and P. Lazaro, “Automated diagnosis for UMTS networks using Bayesian network approach,” *IEEE Transactions on Vehicular Technology*, vol. 57, no. 4, pp. 2451–2461, July 2008.

[54] S. Rezaei, H. Radmanesh, P. Alavizadeh, H. Nikoofar, and F. Lahouti, “Automatic fault detection and diagnosis in cellular networks using operations support systems

data," in *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS 2016)*, April 2016, pp. 468–473.

[55] E. J. Khatib, R. Barco, A. Gómez_Andrades, and I. Serrano, "Diagnosis based on genetic fuzzy algorithms for LTE self_healing," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 3, pp. 1639–1651, March 2016.

[56] G. Ciocarlie, U. Lindqvist, K. Nitz, S. Nováczki, and H. Sanneck, "On the feasibility of deploying cell anomaly detection in operational cellular networks," in *2014 IEEE Network Operations and Management Symposium (NOMS 2014)*, May 2014, pp. 1–6.

[57] Q. Liao and S. Stanczak, "Network state awareness and proactive anomaly detection in self_organizing networks," in *2015 IEEE Globecom Workshops (GC Wkshps)*, Dec 2015, pp. 1–6.

[58] A. Terra, R. Inam, S. Baskaran, P. Batista, I. Burdick, and E. Fersman, "Explainability Methods for Identifying Root-Cause of SLA Violation Prediction in 5G Network," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–7.

[59] S. Tang, J. Kong, B. Niu, and Z. Zhu, "Programmable Multilayer INT: An Enabler for AI-Assisted Network Automation," *IEEE Communications Magazine*, vol. 58, no. 1, pp. 26–32, 2020.

[60] Y. Wei, M. Peng, and Y. Liu, "Intent-based networks for 6G: Insights and challenges," *Digital Communications and Networks*, vol. 6, no. 3, pp. 270–280, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352864820302418

[61] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran_Gia, and R. Schatz, "YoMoApp: A tool for analyzing QoE of YouTube HTTP adaptive streaming in mobile networks," in *European Conference on Networks and Communications (EuCNC)*, 2015.

[62] D. Merkel, “Docker: Lightweight Linux Containers for Consistent Development and Deployment,” *Linux J.*, vol. 2014, no. 239, Mar. 2014.

[63] A. Finamore, M. Mellia, M. Meo, M. M. Munafo, P. D. Torino, and D. Rossi, “Experiences of internet traffic monitoring with tstat,” *IEEE Network*, vol. 25, no. 3, pp. 8–14, May 2011.

[64] P. Casas, P. Fiadino, S. Wassermann, S. Traverso, A. D’Alconzo, E. Tego, F. Matera, and M. Mellia, “Unveiling network and service performance degradation in the wild with mplane,” *IEEE Communications Magazine*, vol. 54, no. 3, pp. 71–79, March 2016.

[65] E. Bocchi, L. De Cicco, and D. Rossi, “Measuring the quality of experience of web users,” *SIGCOMM Comput. Commun. Rev.*, vol. 46, no. 4, pp. 8–13, Dec. 2016.

[66] A. Schwind, M. Seufert, O. Alay, P. Casas, P. Tran_Gia, and F. Wamser, “Concept and Implementation of Video QoE Measurements in a Mobile Broadband Testbed,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[67] I.-R. Learmonth, B. Trammell, M. Kühlewind, and G. Fairhurst, “PATHspider: A tool for active measurement of path transparency,” in *First ACM/IRTF Applied Networking Research Workshop*, Berlin, Germany, Jul 2016.

[68] H. Bai and M. Atiquzzaman, “Error modeling schemes for fading channels in wireless communications: A survey,” *IEEE Communications Surveys Tutorials*, vol. 5, no. 2, pp. 2–9, Fourth 2003.

[69] A. Safari Khatouni, M. Mellia, M. Ajmone Marsan, S. Alfredsson, J. Karlsson, A. Brunström, O. Alay, C. M. A. Lutu, and V. Mancuso, “Speedtest_like Measurements in 3G/4G Networks: The MONROE Experience,” in *Proc. of ITC29*, 2017.

[70] O. Alay, A. Lutu, M. Peón-Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunström, A. Safari Khatouni, M. Mellia, and

M. Ajmone Marsan, “Experience: An Open Platform for Experimentation with Commercial Mobile Broadband Networks,” in *Proc. of ACM Mobicom.*, 2017.

[71] P. Sutton and I. Gomez, “MONROE_SOPHIA _ A Software Radio Platform for Mobile Network Measurement,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[72] P. Torres, P. Marques, H. Marques, R. Dionísio, T. Alves, L. Pereira, and J. Ribeiro, “Data Analytics for Forecasting Cell Congestion on LTE Networks,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[73] G. Aceto, V. Persico, A. Pescapé, and G. Ventre, “SOMETIME: SOftware defined network_based Available Bandwidth MEasuremenT In MONROE,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[74] I. Alepuz, J. Cabrejas, J. Monserrat, A. Perez, G. Pajares, and R. Gimenez, “Use of Mobile Network Analytics for Application Performance Design,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[75] A. Custura, A. Venne, and G. Fairhurst, “Exploring DSCP modification pathologies in mobile edge networks,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[76] I. Learmonth, A. Lutu, G. Fairhurst, D. Ros, and O. Alay, “Path Transparency Measurements from the Mobile Edge with PATHspider,” in *Proc. of the IEEE/IFIP Workshop on Mobile Network Measurement*, Jun. 2017.

[77] A. Amirante, T. Castaldi, L. Miniero, and S. P. Romano, “On the seamless interaction between WebRTC browsers and SIP_based conferencing systems,” *IEEE Communications Magazine*, vol. 51, no. 4, pp. 42–47, April 2013.

[78] E. Bertin, S. Cubaud, S. Tuffin, N. Crespi, and V. Beltran, “WebRTC, the day after: What’s next for conversational services?” in *2013 17th International Conference on Intelligence in Next Generation Networks (ICIN)*, Oct 2013, pp. 46–52.

[79] A. Amirante, T. Castaldi, A. Gouaillard, L. Miniero, S. G. Murillo, and S. P. Romano, “Bringing privacy to the Janus WebRTC server: The PERC way,” in *2017 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, Sept 2017, pp. 1–8.

[80] M. V. data set. (2018). [Online]. Available: https://doi.org/10.5281/zenodo.1188410

[81] M.-W. Container. (2018). [Online]. Available: https://github.com/acmmmsys/2018-MONROE-webstreamer/

[82] K. Nepomuceno, I. N. d. Oliveira, R. R. Aschoff, D. Bezerra, M. S. Ito, W. Melo, D. Sadok, and G. Szabó, “QUIC and TCP: A performance evaluation,” in *IEEE ISCC*, June 2018.

[83] P. Qian, N. Wang, and R. Tafazolli, “Achieving robust mobile Web content delivery performance based on multiple coordinated QUIC connections,” *IEEE Access*, vol. 6, pp. 11 313–11 328, 2018.

[84] R. Lychev, S. Jero, A. Boldyreva, and C. Nita_Rotaru, “How secure and quick is QUIC? Provable security and performance analyses,” in *IEEE Symposium on Security and Privacy*, 2015.

[85] M. Bishop, “Hypertext transfer protocol version 3 (HTTP/3),” Internet Engineering Task Force, Internet-Draft draft-ietf-quic-http-34, Feb. 2021, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-ietf-quic-http-34

[86] R. Marx, “QUIC and HTTP/3 event definitions for qlog,” Internet Engineering Task Force, Internet-Draft draft-marx-qlog-event-definitions-quic-h3-02, Nov. 2020, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-marx-qlog-event-definitions-quic-h3-02

[87] ——, “Main logging schema for qlog,” Internet Engineering Task Force, Internet-Draft draft-marx-qlog-main-schema-02, Nov. 2020, Work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/html/draft-marx-qlog-main-schema-02

[88] V. Arun and H. Balakrishnan, “Copa: Practical delay_based congestion control for the Internet,” in *Proceedings of ANRW*, 2018.

[89] P. Yang, Y. Xiao, M. Xiao, and S. Li, “6G wireless communications: Vision and potential techniques,” *IEEE Network*, vol. 33, no. 4, pp. 70–75, July 2019.

[90] M. Rajiullah, A. Lutu, A. S. Khatouni, M.-R. Fida, M. Mellia, A. Brunstrom, O. Alay, S. Alfredsson, and V. Mancuso, “Web experience in mobile networks: Lessons from two million page visits,” in *The World Wide Web Conference*, ser. WWW ’19. New York, NY, USA: Association for Computing Machinery, 2019, p. 1532–1543. [Online]. Available: https://doi.org/10.1145/3308558.3313606

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

[92] S. Biaz and N. H. Vaidya, “Is the round_trip time correlated with the number of packets in flight?” in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement IMC’03.* New York, NY, USA: Association for Computing Machinery, 2003. [Online]. Available: https://doi.org/10.1145/948205.948240

[93] S. Dhanorkar, C. T. Wolf, K. Qian, A. Xu, L. Popa, and Y. Li, “Who Needs to Know What, When?: Broadening the Explainable AI (XAI) Design Space by Looking at Explanations Across the AI Lifecycle,” in *Designing Interactive Systems Conference 2021*, ser. DIS ’21. New York, NY, USA: Association for Computing Machinery, 2021, p. 1591–1602. [Online]. Available: https://doi.org/10.1145/3461778.3462131

[94] M. Langer, D. Oster, T. Speith, H. Hermanns, L. Kästner, E. Schmidt, A. Sesing, and K. Baum, “What do we want from Explainable Artificial Intelligence (XAI)? – A stakeholder perspective on XAI and a conceptual model guiding interdisciplinary

XAI research," *Artificial Intelligence*, vol. 296, p. 103473, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370221000242

[95] N. Bostrom and E. Yudkowsky, "The ethics of artificial intelligence," *The Cambridge handbook of artificial intelligence*, vol. 1, pp. 316–334, 2014.

[96] O. Alay, A. Lutu, M. Peón_Quirós, V. Mancuso, T. Hirsch, K. Evensen, A. Hansen, S. Alfredsson, J. Karlsson, A. Brunstrom, A. Safari Khatouni, M. Mellia, and M. A. Marsan, "Experience: An open platform for experimentation with commercial mobile broadband networks," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 70–78. [Online]. Available: https://doi.org/10.1145/3117811.3117812

[97] P. D. Grünwald, *The Minimum Description Length Principle (Adaptive Computation and Machine Learning).* The MIT Press, 2007.

[98] C. S. Wallace and D. L. Dowe, "Minimum Message Length and Kolmogorov Complexity," *The Computer Journal*, vol. 42, no. 4, pp. 270–283, 01 1999. [Online]. Available: https://doi.org/10.1093/comjnl/42.4.270

[99] Y. Yang and G. I. Webb, "Discretization for naive_bayes learning: managing discretization bias and variance," *Machine learning*, vol. 74, no. 1, pp. 39–74, 2009.

[100] S. García, J. Luengo, J. A. Sáez, V. López, and F. Herrera, "A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 4, pp. 734–750, 2013.

[101] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, "Machine learning interpretability: A survey on methods and metrics," *Electronics*, vol. 8, no. 8, p. 832, 2019.

[102] L. Breiman, J. Friedman, R. Olshen, and C. Stone, "Classification and regression trees." *Kluwer Academic Publishers, New York*, 1984.

[103] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer, 2009. [Online]. Available: http://www-stat.stanford.edu/~tibs/ElemStatLearn/

[104] W. Loh, "Fifty years of classification and regression trees," *International Statistical Review*, vol. 82, no. 3, pp. 329–348, 2014.

[105] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of max_dependency, max_relevance, and min_redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 8, p. 1226–1238, Aug. 2005. [Online]. Available: https://doi.org/10.1109/TPAMI.2005.159

[106] G. Brown, A. Pocock, M. Zhao, and M. Luján, "Conditional likelihood maximisation: A unifying framework for information theoretic feature selection," *J. Mach. Learn. Res.*, vol. 13, no. null, p. 27–66, Jan. 2012.

[107] R. Cilibrasi and P. M. Vitányi, "Clustering by compression," *IEEE Transactions on Information theory*, vol. 51, no. 4, pp. 1523–1545, 2005.

[108] D. C. Hoaglin, F. Mosteller, and J. W. T. (Editor), *Understanding Robust and Exploratory Data Analysis*, 1st ed. Wiley_Interscience, 2000.

[109] D. Arthur and S. Vassilvitskii, "K_means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM_SIAM Symposium on Discrete Algorithms*, ser. SODA '07. USA: Society for Industrial and Applied Mathematics, 2007, p. 1027–1035.

[110] P. M. B. Vitányi, F. J. Balbach, R. Cilibrasi, and M. Li, "Normalized information distance," *CoRR*, vol. abs/0809.2553, 2008. [Online]. Available: http://arxiv.org/abs/0809.2553

[111] V. Mancuso, M. Peón Quirós, C. Midoglu, M. Moulay, V. Comite, A. Lutu, O. Alay, S. Alfredsson, M. Rajiullah, A. Brunström, M. Mellia, A. Safari Khatouni, and T. Hirsch, "Results from running an experiment as a service platform for

mobile broadband networks in europe," *Computer Communications*, vol. 133, pp. 89–101, 2019. [Online]. Available: http://infoscience.epfl.ch/record/257228

[112] H. Xiong, J. Wu, and J. Chen, "K-means clustering versus validation measures: A data-distribution perspective," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, pp. 318 – 331, 05 2009.

[113] S. Sieranoja, "K-means properties on six clustering benchmark datasets," *Applied Intelligence*, vol. 48, 12 2018.

[114] C. M. Bishop, "Mixture density networks," 1994.

[115] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978. [Online]. Available: http://www.jstor.org/stable/2958889

[116] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*, 1st ed. Springer, 2007. [Online]. Available: http://www.amazon.com/Pattern-Recognition-Learning-Information-Statistics/dp/0387310738%3FSubscriptionId%3D13CT5CVB80YFWJEPWS02%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0387310738