Control Principles for Autonomous Communication Networks

Azza H. Ahmed

OsloMet Avhandling 2023 nr 23

OSLO METROPOLITAN UNIVERSITY
STORBYUNIVERSITETET

Control Principles for Autonomous Communication Networks

Azza H. Ahmed

PhD program in Engineering Science
Department of Computer Science
Faculty of Technology, Art and Design
OsloMet – Oslo Metropolitan University

Spring 2023

OsloMet Avhandling 2023 nr 23

ISSN 2535-471X (trykt)
ISSN 2535-5414 (online)
ISBN 978-82-8364-490-6 (trykt)
ISBN 978-82-8364-506-4 (online)

OsloMet – storbyuniversitetet
Universitetsbiblioteket
Skriftserien
St. Olavs plass 4,
0130 Oslo,
Telefon (47) 64 84 90 00

Postadresse:
Postboks 4, St. Olavs plass
0130 Oslo

Trykket hos Byråservice

Trykket på Scandia 2000 white, 80 gram på materiesider/200 gram på coveret

Preface

Advancements in artificial intelligence (AI) and network softwarization have enabled the automation of complex networking problems that were previously difficult to solve. The work presented in this thesis explores the use of AI techniques for solving various networking problems, including network anomalies prediction, detection and attribution of network outages and network optimization. I carried out this research under the supervision of Dr. Ahmed Elmokashfi and Professor Michael A. Riegler at Simula Metropolitan Center for Digital Engineering (SimulaMet).

This dissertation comprises five articles, which have been published or is under review in leading networking and machine learning conferences and journals. Since the initial submission of this thesis, Article II, Article III, Article IV have been published in conferences and journals, while Article V is currently under review at IEEE Transactions on Mobile Computing. The articles included in this thesis have been printed in their initial versions. Links to the final published versions can be found at the following URLs:

Article I: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9663160
Article II: https://dl.acm.org/doi/pdf/10.1145/3534678.3539097
Article III: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9831432
Article IV: https://dl.acm.org/doi/pdf/10.1145/3547115.3547193
Article V: https://www.techrxiv.org/articles/preprint/Bottleneck_Identification_in_Cloudified_Mobile_Networks_based_on_Distributed_Telemetry/22100546(Preprint)

I hope that this research will contribute to the ongoing efforts to leverage AI in solving networking problems and that it will inspire further innovation in this field.

Azza H. Ahmed
April, 2023
Oslo Norway

To the soul of my sincere friend, my father who could not see this thesis completed.

Abstract

The growing complexity of communication networks and the explosion of network traffic have made the task of managing these networks exceedingly hard. A potential approach for striking this increasing complexity is to build an autonomous self-driving network that can measure, analyze and control itself in real time and in an automated fashion without direct human intervention. In this thesis, we focus on realizing such an autonomous network leveraging state-of-the-art networking technologies along with artificial intelligence and machine learning techniques. Toward this goal, we exploit different learning paradigms to automate network management. First, we propose supervised machine learning methods to detect increases in delays in mobile broadband networks. Further, considering the challenges of supervised learning in networking applications, we present a novel real-time distributed architecture for detecting anomalies in mobile network data in an unsupervised fashion. It also involves a collaborative framework for knowledge sharing between the distributed probes in the network to improve the overall system accuracy. Second, we propose a novel deep reinforcement learning based control framework for optimizing resources utilization while minimizing performance degradation in multi-slice Radio Access Network (RAN) through a set of diverse control actions. We explore both centralized and distributed control architectures. Last, we design a framework for timely collecting telemetry, detecting and attributing outages in mobile networks. We evaluate our framework on a software defined virtualised testbed that resembles a cloudified mobile network.

Sammendrag

Sammendrag Den økende kompleksiteten til kommunikasjonsnettverk og eksplosjonen av nettverkstrafikk har gjort oppgaven med å administrere slike nettverk stadig vanskeligere. En potensiell tilnærming for å møte denne økende kompleksiteten er å bygge et autonomt selvkjørende nettverk som kan måle, analysere og kontrollere seg selv i sanntid på en automatisert måte uten direkte menneskelig innblanding. I denne avhandlingen fokuserer vi på å realisere slike autonome nettverk som utnytter moderne nettverksteknologier sammen med kunstig intelligens og ulike teknikker for maskinlæring. For å realisere dette utnytter vi forskjellige maskinlæringsparadigmer til å automatisere nettverksadministrasjon. Først foreslår vi metoder med veiledet maskinlæring for å oppdage økninger i forsinkelser i mobile bredbåndsnettverk. Videre, med tanke på utfordringene med veiledet læring i nettverksapplikasjoner, presenterer vi en ny distribuert sanntidsarkitektur for å oppdage uregelmessigheter i data fra mobilnettverk uten å bruke veiledet maskinlæring. Det innebærer også et samarbeidsrammeverk for kunnskapsdeling mellom distribuerte prober i nettverket for å forbedre den generelle systemets nøyaktighet. For det andre foreslår vi et nytt, dypt forsterkende læringsbasert kontrollrammeverk for å optimalisere ressursutnyttelsen og samtidig minimere ytelsesdegradering i multi-slice Radio Access Network (RAN) gjennom et sett med forskjellige kontrollhandlinger. Vi utforsker både sentraliserte og distribuerte kontrollarkitekturer. Til slutt presenterer vi et rammeverk for sanntids innsamling av telemetri som kan oppdage og identifisere utfall i mobilnettverk. Vi evaluerer rammeverket vårt i en programvaredefinert virtualisert infrastruktur som tilsvarer et skybasert mobilnettverk.

Acknowledgements

This PhD journey has allowed me to meet exciting people who have greatly helped and supported me. These people have allowed me to grow, learn, and progress both scientifically and personally. I would like to acknowledge those who have contributed the most throughout this journey.

First, I would like to thank my main supervisor Ahmed Elmokashfi for his support and his immense scientific knowledge. He has been guiding me throughout this journey with his inspiring advices and constructive criticism. I really appreciate his patience, availability and trust. Without his precious contributions it would not be possible to complete this work. I also would like to thank Michael Riegler for being my co-supervisor and giving me a thorough introduction to the research area of artificial intelligence.

I am also very grateful to all my colleagues at SimulaMet, specially to MahRukh Fida, Andres Ocampo, Steven Hicks, Anas Al-Selwi, Ioana Livadariu, Foivos Michelinakis, Thomas Dreibholz, Jan Marius Evang, and Haakon Bryhni for their valuable knowledge, strong encouragement, and for all the fun we have had together in the last three years.

Finally, I express my gratitude to my family and friends for all of the love and support along this journey. To my parents, thank you for being my champions throughout the past 33 years. To my brother, thank you for believing in me. To my children; Noor, Mohamed and Yousif, you are my inspiration to achieve greatness. To my friends; Lina, Sara, Samah, Mohamed and Rasha, you have made me stronger, better and more fulfilled than I could have ever imagined. Last but not least, special thanks to my husband Sami Satti, for his love, his patience, and his choice to share with me the happy and the difficult moments of the PhD journey.

List of Articles

Article I Ahmed, A. H., Hicks, S., Riegler, M. A., and Elmokashfi, A. (2021). **Predicting High Delays in Mobile Broadband Networks.** IEEE Access, 9, 168999-169013. DOI: https://doi.org/10.1109/ACCESS.2021.3138695.

Article II Ahmed, A. H., Hicks, S., Riegler, M. A., and Elmokashfi, A. (2022, August 14 - 18,) **RCAD:Real-time Collaborative Anomaly Detection System for Mobile Broadband Net-works**. KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. DOI: https://doi.org/10.1145/3534678.3539097

Article III Ahmed, A. H., and Elmokashfi, A. **ICRAN: Intelligent Control for Self-driving RAN based on Deep Reinforcement Learning**, in IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 2751-2766, Sept. 2022.
DOI: https://doi.org/10.1109/TNSM.2022.3191746.

Article IV Evang, J. M., Ahmed, A. H., Elmokashfi, A. and Bryhni, H. (2022, 26. July) **Crosslayer Network Outage Classification Using Machine Learning.** Applied Networking Research Workshop 2022 (ANRW'22).
DOI: https://doi.org/10.1145/3547115.3547193

Article V Fida*, M., Ahmed*, A. H., Dreibholz, T., Ocampo, A. F., Michelinakis, F. I and Elmokashfi, A. **Bottleneck identification in cloudified mobile networks based on distributed telemetry.** Submitted to Conference on emerging Net-working EXperiments and Technologies (CoNEXT 2022).URL: https://www.techrxiv.org/articles/preprint/Bottleneck_Identification_in_Cloudified_Mobile_Networks_based_on_Distributed_Telemetry/22100546

* Both authors contributed equally.

Part I: Research Overview

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The basic functioning of our society has become increasingly reliant on communication infrastructures ranging from the Internet to mobile networks. Next-generation mobile networks has transformed the mobile network into a multi-service architecture that supports diverse use cases with varying requirements [33]. Today, public safety, healthcare, and businesses require reliable and robust communication networks. Moreover, the rise of the Internet of Things (IoT) and the need to tackle pressing societal challenges such as population aging and global warming, mean that our reliance on communication infrastructures is only going to rise. According to the International Telecommunication Union (ITU), the overall mobile data traffic is estimated to grow at an annual rate of around 55% in 2020–2030 to reach 607 exabytes (EB) per month in 2025 and 5,016 EB in 2030 [1]. Overall, this rapid increase and change in the number of connected devices, applications and data volume in networks are putting a significant pressure on the current network management approaches that heavily rely on human operators. Furthermore, several envisioned use cases like public safety communication and industrial control have stringent performance expectations [49]. Because of this growing need for diverse applications, each with its own quality of service (QoS) requirements, today's operators must maintain a balance between the the provided services and the reduction of configuration problems. In addition, this diversity increases security, availability, and performance requirements of these applications which makes network management a difficult task in real time across a complex web of interacting protocols and systems [30]. Several studies show that a

considerable fraction of network failures is related to human errors while performing network management tasks. For example, Liu et al. [72] investigated the network incidents in Alibaba wide area network (WAN), that took place in the years 2016 and 2017, and found that 56% of them were due to misconfigurations. A similar study from Google, which analyzed failures in Google's infrastructure, concluded that most of the failures occurred during the network management process [38]. Recently, in October 2021, Facebook experienced an outage that lasted for six hours which left over 3.5 billion users with no access to its social media platforms (Facebook, Instagram and WhatsApp). According to Facebook's official incident report [25], the outage was caused by a maintenance routine job which unintentionally took down all the connections in Facebook's backbone network, effectively disconnecting Facebook data centers globally. Moreover, in December 2021, a major outage hit Amazon Web Services (AWS) in Northern Virginia (US-EAST-1) region affecting most of Amazon services such as Alexa, Ring, and Disney Plus. Amazon explained that the failure was due to an automated activity to scale capacity of one of the AWS services hosted in the main AWS network which triggered an unexpected behavior from a large number of clients inside the internal network. Due to the high volume of connections as a result, the internal network and the main AWS network's networking hardware became overloaded, which delayed communication between both networks [8].

The requirement to meet the needs of both users and operators in a cost-effective way has motivated the academia and industry to argue for building "autonomous" or "self-driving" networks. A self-driving network is defined as a network capable of measuring, analyzing and controlling itself where network management and control decisions are made in real time and in an automated fashion with little or no human involvement. The campaign for more autonomous networks has been accompanied by the recent technological advances mainly: 1) the development of fully programmable data planes and the languages to program them; and 2) the breakthroughs in Artificial Intelligence (AI) and Machine Learning (ML) algorithms [30].

In response to this interest within academia and industry in leveraging the latest technologies to advance the development of autonomous networks, in this work we exploit the advantages of ML in solving complex performance issues in networks, especially with the advancement of the future network such as software-defined networks, network virtualization and network slicing. Further, existing approaching for closed-loop control in

communication networks are mostly centralized and threshold-based and are thus rigid. Hence, in this work we tackle the challenge of the distributed nature of communication networks and the need of executing decision making processes over very short time scales to realize the self-driving automated future networks. We present an automated network management system that can efficiently predict and detect anomalies in real time. Furthermore, we propose a novel intelligent control method that can timely detect and correct performance problems. Specifically, we focus on maximizing network radio resources utilization and minimizing performance degradation by leveraging the advances in deep reinforcement learning. We argue that building self-driving networks is feasible if we undertake transformation from passive response to proactive prediction and from open-loop to closed-loop control. The integration of AI and networking technologies such as programmablility and virtualization makes this goal seemly achievable.

1.2 Thesis Scope and Research Questions

Figure 1.1 presents the scope of this PhD thesis. The core contributions of the thesis are related to three domains representing the main components of self-driving networks: telemetry, data analytic and network control and management. The main research question of this study is:

Can AI-based methods help in realizing self-driving networks?

To answer this research question, we analyze the related state-of-the-art and formulate the following detailed three research questions (RQs). For each RQ, we refer to our main contributions by pointing to the respective paper (or collection of papers) that addresses the research problem.

RQ 1: How to build an automated network management system that can efficiently predict and detect anomalies in network traffic?

The need for automated management system is highly driven by the rapid increase in data traffic and the number of connected devices and applications in networks which are putting a significant pressure on the current network management approaches that heavily rely on human operators. One of the main challenges to network management systems is "anomalies" which represent unusual traffic patterns in the network. To help identify these

anomalies, we leverage different AI/ML methods. **Article I** proposes a supervised learning ensemble classification model that predicts high delays in mobile broadband networks. In this study, we consider 4G and 5G measurements from the two largest mobile operators in Norway. The probes measure round trip times (RTTs) alongside connection metadeta. In a different context other than the mobile networks, we conduct another study based on supervised learning in **Article IV** to identify and classify crosslayer outages in a global Internet Service Provider (ISP) network using Layer1, Layer2 and Layer3 data. Further, to tackle the supervised learning challenges we leverage unsupervised learning for detecting anomalies in networks. In particular, in **Article II** we propose a novel unsupervised framework for detecting anomalies in network data forwarding latency in an distributed way. In this work, we use Hierarchical Temporal Memory (HTM) model to detect anomalies in real time in a commercial mobile network. We also develop a novel collaborative distributed learning module that facilitates knowledge sharing across the system to improve the overall system performance.

RQ 2: How to design an intelligent control framework for future networks that maximizes the network resources and assure users' satisfaction? Is it possible to increase control decisions' efficiency by incorporating learning into network management and control? What are the trade-offs between centralized and decentralized control architectures in network management?

Mobile networks are increasingly expected to accommodate a wide range of use cases with varying performance requirements while maintaining a high level of reliability. These expectations necessitate methods for detecting and correcting performance issues in a timely manner. Current approaches, on the other hand, frequently focus on optimizing a single performance metric. To address this gap, we present, in **Article III**, a novel dual-objective control framework that maximizes radio resources utilization while minimizes performance degradation in radio access network (RAN) which represents the most challenging part of cellular architecture. We leverage two deep reinforcement learning based approaches that control the RAN in a centralized and a distributed way, respectively, through a set of diverse control actions. To evaluate our proposed methods, we use the well-known network simulator ns-3 [2].

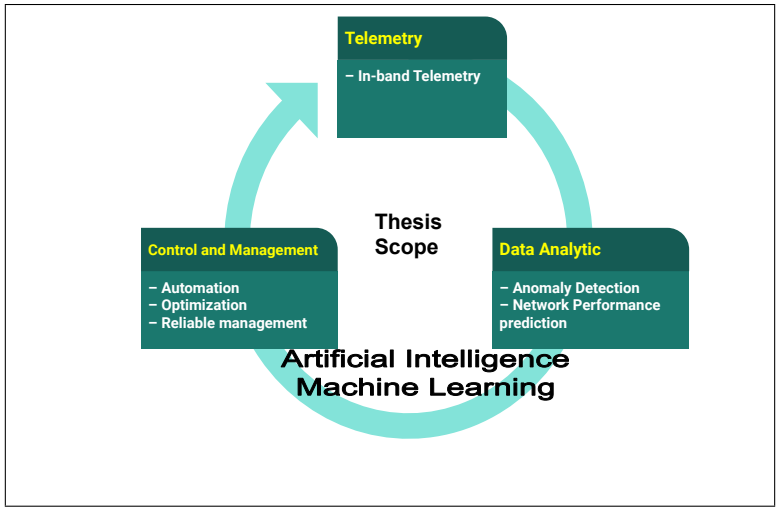


Figure 1.1: Thesis Scope.

RQ 3: How to design and implement a troubleshooting system that can detect performance degradation problems in mobile network leveraging the advances in AI/ML based on the network telemetry? What are the minimal measurements should be collected to identify the root cause of such problems?

Next-generation mobile networks are expected to support different services with different requirements across heterogeneous network components and virtualized infrastructures. To meet the predefined service level agreement (SLA), for an optimal user experience, it is vital for the operators to constantly monitor various points in the network. It is required to identify faults in the network and resolve the issues in a timely manner, so as to minimize service downtime. In addition to infrastructure performance, it is essential to monitor applications from an end user perspective. Therefore, in **Article V** we implement a distributed telemetry system on a real testbed in which end users act as early warning sensors. Upon flagging a performance degradation problem, the network controller activates its machine learning model to attribute the cause of the problem with minimal system overhead.

1.3 Research Methodology

In this work, we follow the below four different research methods in order to answer the aforementioned research questions:

- Measurements and empirical data being used in this work are collected from two different sources: 1) NorNet Edge (NNE) platform [60]; a large number of probes setup for measuring commercial mobile broadband networks in Norway, and 2) extensive measurements in a dedicated experimental setup, i.e., testbeds.

- Statistical methods to provide data analysis and preliminary insights into the problem under investigation. We leverage both descriptive statistics such as mean and standard deviation, and inferential statistics such as hypothesis testing and regression analysis.

- Machine learning and biologically-inspired machine intelligence algorithms are leveraged and optimized to capture the intrinsic characteristics of the network data. Different learning algorithms are used in this work: supervised [19], unsupervised [123], deep learning [62], deep reinforcement learning [68] and HTM [44].

- Network simulation offers an efficient cost-effective way to assess how the network will behave under different conditions. We use the ns-3 simulator [2]; which is an open source discrete-event network simulator; to create a training environment for deep reinforcement learning (DRL) agents. More specifically, we use ns-3 Gym [36] which is an interface between OpenAI Gym and ns-3 that allows for integration of those two frameworks. The interface takes care of the management of the ns-3 simulation process life cycle as well as delivering state and action information between the DRL agent and the simulation environment. We choose simulation for training because DRL agents need a huge number of iterations to properly learn. Also, the simulation allows DRL agents to explore different actions without affecting the functionalities of network hardware components. However, we believe that an agent trained by simulation data needs further adaptation before transfer to the real environment.

1.4 Thesis Outline

This thesis is organized into two main parts. Part I outlines the research and identifies the contributions of the thesis. It is divided into four chapters: *Chapter1- Introduction* (this chapter) presents the motivation for this work, research questions addressed in this

thesis and the research methods, *Chapter 2-Background* gives background information on automated network management and control alongside machine learning algorithms applied in this work, *Chapter 3- Related Work and Research Contributions* summarizes the related work and identifies the thesis contributions, and finally *Chapter4- Conclusion and Future Work* concludes the thesis and discusses potential future work directions. Part II contains each of the five papers published during the thesis in their original shapes. Below is the list of these publications:

Paper I	Predicting High Delays in Mobile Broadband Networks.
published in	IEEE Access, Volume 9, pages 168999-169013, 2021.
authors	Azza H. Ahmed, Steven Hicks, Michael A. Riegler, and Ahmed Elmokashfi.
Paper II	RCAD:Real-time Collaborative Anomaly Detection System for Mobile Broadband Networks.
accepted at	ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2022.
authors	Azza H. Ahmed, Michael A. Riegler, Steven Hicks and Ahmed Elmokashfi.
Paper III	ICRAN: Intelligent Control for Self-driving RAN based on Deep Reinforcement Learning.
submitted to	IEEE Transactions on Network and Service Management Journal, 2022.
authors	Azza H. Ahmed and Ahmed Elmokashfi.
Paper IV	Crosslayer Network Outage Classification Using Machine Learning.
accepted at	ACM/IRTF Applied Networking Research Workshop (ANRW'22), 2022.
authors	Jan Marius Evang, Azza H. Ahmed, Ahmed Elmokashfi and Haakon Bryhni.
Paper V	Bottleneck identification in cloudified mobile networks based on distributed telemetry.
submitted to	Conference on emerging Networking EXperiments and Technologies (CoNEXT 2022).

authors	MahRukh Fida*, Azza H. Ahmed*, Thomas Dreibholz, Andres F. Ocampo, Foivos A. Michelinakis and Ahmed Elmokashfi.

* Both authors contributed equally.

Chapter 2

Background

In this chapter, we introduce some background information for a better understanding of this PhD thesis. In Section 2.1, we focus on the new network management paradigm, namely self-driving networks. In Section 2.2, we present the machine learning advances in the context of network management.

2.1 Self-driving Networks

The network management automation is vital to meet the increasing complexity in today's communication infrastructures. For example, 5G has transformed the mobile network into a multi-service architecture that supports diverse use cases with varying requirements [33]. 5G virtualizes network resources and chains them into end-to-end network slices that are adapted to use cases' requirements. This flexibility makes mobile networks increasingly complex to manage [83]. Hence, multi-slice 5G and beyond networks must be capable of quickly detecting and correcting performance degradation. Current mobile networks resort to pre-configured priorities, over-provisioning and at best implementing traditional closed loop control systems with a limited scope like in the case of self-organizing networks (SON) [113]. While most of the research in autonomous networks has been focused on mobile networks, recently there has been an increased interest in network management automation for other networks such as wide area network (WAN) which represents one of the most significant networks in the Internet. Consequently, Software-defined WAN (SD-WAN) is presented as promising architecture of next-generation WAN to address the challenges of the increasing number of devices connected to the Internet and traffic

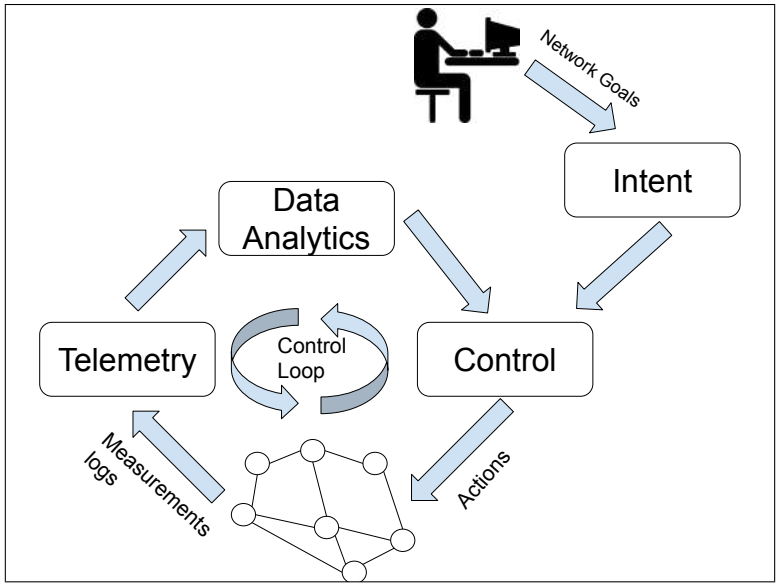


Figure 2.1: Self-Driving Networks High-Level Architecture.

demands [133]. Compared with traditional WAN, SD-WAN provides a programmable infrastructure to develop centralized control to define network policies, monitor network traffic, configure network devices and assure users' quality of experience without human intervention [82].

The need for intelligent automation has motivated the academia and industry to argue for building "autonomous" or "self-driving" networks, where network management and control decisions are made in real time and in an automated fashion. A self-driving network is defined as a network capable of measuring, analyzing and controlling itself in an automated manner [54]. Figure 2.1 shows the high-level architecture proposed by both the academia and the industry for self-driving networks. In this architecture, the self-driving network can take as input high-level goals related to performance or security (such as minimizing network congestion) through the Intent-based system and jointly determine (1) the measurements that the network should collect, (2) learning and inferences that the network should perform, and (3) the actions that the network should ultimately execute [30]. Below, we will deep dive into those components individually and identify the enabling technologies to realizing self-driving networks.

2.1.1 Main Components

(a) ***Telemetry:*** One of the ultimate goals of the self-driving networks is reducing the operational and maintenance costs result from the growing complexity of the

networks. The network management system should scale to a large amount of traffic generated by the huge number of networked devices. This implies revisiting network telemetry. Flow monitoring tools like NetFlow [3], Flowradar [67] and sFlow [128] provide good performance for coarse-grained tasks, but with high memory and bandwidth consumption. Operators collect massive amount of data from these tools and infer network-wide state from the collected data. Aligned with the vision of the self-driving networks, the network telemetry system should monitor network flows, analyze the measurements and provide the needed information about network state to the network management system to make the right management decisions in a real-time and fine-grained fashion [134]. Moreover, through the network telemetry system the operators should be able to specify high-level queries about the network-wide state without worrying how the measurements are conducted. Finally, telemetry should provide better scalability, accuracy, coverage, and performance than traditional network measurement technologies [132]. With the rise of software defined networking (SDN) and programmable data plane, different telemetry approaches are being proposed. For example, NetSight [43] which applies packet monitoring to support applications that require extracting the full journey of selected packets. Based on the packet history, each application can analyze the collected data of interest and answer the network-wide queries. Another work that rely on packet-level telemetry is Everflow [137] which is used in large data centres. Everflow traces specific packets by applying match+action functionality to filter packets. Both NetSight and Everflow need large bandwidth and processing overheads when the network is at a large scale. Recently, a new telemetry concept has been introduced, In-band Network Telemetry (INT) [57] which relies on programmable switches to execute queries. It encapsulate packet-level network state (e.g., link utilization, queue occupancy, hop-by-hop delays) into "production traffic" packets. INT allows collecting and reporting network states in the data plane, without intervention from the control plane. Several recent works have explored the idea of INT. The P4-based INT [23] is the earliest INT implementation using programmable switches. Pan et al. [90] proposed INT-path; a framework comprises of INT and active telemetry. INT-path couples the INT probe with the source routing label stack to accommodate the user-specified monitoring path. Snappy [22]

also utilizes the data-plane memory to maintain multiple snapshots of the queue occupancy to identify the flows responsible for a microburst in real time. Although INT emerges as a promising telemetry approach to provide fine-grained data, it can result in performance degradation due to encapsulation of telemetry instruction and metadata data. The control unit should optimize between the INT overhead and the achievements of the wide-network goals.

(b) ***Data Analytic:*** The telemetry system collects large monitoring and measurement datasets that must be aggregated, filtered, and analyzed. AI and ML offer techniques for extracting knowledge from data [9]. They can play a vital role in facilitating closed loop orchestration and management to realize automated networks. Self-driving network leverages AI/ML techniques to extract knowledge and inference from operational data gathered in the telemetry system. Examples of inference include forecasting traffic volume changes [126], estimating user mobility patterns [45], predicting future network events such as throughput drop [97], and early detection of anomalous behavior [64]. These inferences will be leveraged by the control function to take more reliable control and management decisions [17].

(c) ***Control and Decision Making:*** Self-driving networks are only feasible with accurate and timely feedback from the network elements. Existing network monitoring solutions increase network overhead dramatically and can only provide delayed experience of the packet, leading to deferred network control decisions. But, prompt responses to network conditions are very critical for self-driving networks. The role of the control component is to monitor the network configurations and status from the network and based on some triggers from the analytics unit can take actions to modify the current configurations on the network.

(d) ***Control Loop:*** In early 2000s, IBM proposed an architecture for autonomic computing [48] by defining MAPE (Monitor, Analyse, Plan, and Execute) control loop over the managed environment. Since then, other works have proposed new solutions based on the MAPE loop. For example, ETSI [27] introduced the Experiential Networked Intelligence through the concept of closed control loops containing monitoring, analysis, policy, execution plus knowledge steps, MAPE-K, with applications in infrastructure management, network operations, and service orchestration and man-

agement. Also, SON leverages closed control loop as a key enabler for management automation to deliver the self-x properties such as self-healing, self-configuration, and self-optimization [113].

Aligned with these proposals, self-driving networks are proposed as a closed-loop system which enables and adapts network polices (goals) to continuously manage the virtual network in an automated manner. When the network experiences a change, information gathered in the telemetry system is packaged as an event. This event is analysed by the analytics unit to assess its significance. Significant events trigger the control component to perform corrective actions, i.e. reconfiguration on the network. In open control loops, at least one of the stages in the loop is manually performed. In contrast, with closed control loops, the operator only defines a goal and once it is configured, the loop runs automatically. In both open and closed control loop the operator may perform some configurations as part of supervision of the loops. Both control loops attempt in controlling the status of the network by trying to keep it as close as possible to an operator specified desired goal.

(e) ***Intent-based Networking:*** Deploying policies in modern enterprise networks poses significant challenges for today's network operators. Since policies typically describe high-level goals or business intents, the operators must perform the complex and error-prone job of breaking each policy down into low-level tasks and deploying them in the physical or virtual devices of interest across the entire network. Recently, intent-based networking (IBN) has been proposed to solve this problem by allowing operators to specify high-level policies that express how the network should behave (e.g., defining goals for quality of service, security, and performance) without having to worry about how the network is programmed to achieve the desired goals and without the knowledge of the low-level configurations in the network [11]. The operators can use high level languages to express their intents that dictate how the network should behave to achieve the specified goals. This area has attracted many researches in academia and industry through the last few years to create new intent-based policy frameworks. There have been various efforts to present several intent languages and compilers to deploy the translated intents in the network [11, 12, 116]. With the the recent advances in machine learning, various works proposed the usage of the natural languages thus eliminating the need for

learning new languages [51, 50, 6]. For example, Jacobs et al. [51] presented a novel intent-refinement architecture that uses deep learning and feedback from the operator to translate the network high-level objectives expressed in natural language into network commands. The operator has the ability to do some changes for the translated intent program until it is finalized. This work incorporates the feedback from the operator which results in improving the performance for the intent management system while avoiding mis-configurations.

2.1.2 Enabling Technologies to Realize Self-driving Networks

In this section we highlight the key advances that provide great opportunities to realize the concept of self-driving network, in particular, the networking technologies: software-defined networking (SDN), network function virtualization (NFV) and network slicing (NS).

(a) ***Software-Defined Networking (SDN)*:** SDN is a paradigm that separates the data plane and the control plane. Conventionally, network devices such as switches and routers have control plane, management plane and data plane whereas in SDN the control plane logic is implemented as a centralized software component that controls the entire network. Forwarding devices could be programmed using an open interface, such as OpenFlow [79] to enable the control plane to control forwarding devices from different hardware and software vendors. Although most of the research in SDN has focused on the control plane programmability, lately, SDN has also introduced opportunities for data plane flexibility and programmability. [13, 53] survey the current advances in programmable software and hardware network devices. They highlighted the architecture of software switches such as Open vSwitch (OVS) [93] and PISCES [109] and the mechanisms to program them like Programming protocol-independent packet processors (P4) [16]. P4 is currently the most widespread abstraction, programming language, and concept for data plane programming. P4 is a high-level language to dictate the forwarding behavior on a variety of switches. In P4, the programmer declares how packets are to be processed, and a compiler generates a configuration for a protocol-independent switch chip or network interface card (NIC). The P4 language has rapidly captured the attention of the networking community.

(b) ***Network Function Virtualization (NFV)*:** The idea behind NFV is to virtualize hardware and networking resources, i.e. computing, storage, and networking resources of commodity hardware [42]. The main architecture for NFV consists of three building blocks: 1) NFV Infrastructure (NFVI) which virtualizes compute, storage, and networking resources, 2) Virtual Network Functions (VNFs) which are the software implementations of network functions deployed on virtual environment, and finally 3) NFV Management and Orchestration (NFV MANO) block that is responsible for managing and orchestrating VNFs and NFVI [28]. VNFs can be chained to form an end-to-end network architecture. With the evolution of the cloud infrastructure, many operators and vendors are embracing cloud native and container-based software [26]. In this context, cloud-native network functions (CNFs) is a way to develop, deploy, run and manage VNFs that exploit the benefits of the cloud computing model. A fundamental principle of a cloud native application is to decompose software into smaller, more manageable pieces. This is usually done through the utilization of a microservice architecture [119].

(c) ***Network Slicing (NS)*:** NS is a way of creating multiple logical networks on the top of shared infrastructure. The idea of NS can be linked to the Infrastructure as a Service (IaaS) cloud computing paradigm, in which several tenants share computing, networking, and storage resources to create various virtual networks over a common infrastructure [4]. In the context of 5G and beyond, NS can support different use cases where each slice can be allocated based on the specific needs of the use case. For example, critical communication such as smart grid metering needs ultra-low latency and high data speed, whereas applications like virtual reality (VR) gaming needs high bandwidth. NS supports these diverse services by efficiently reassigns virtual resources from one network slice to another while ensuring slices isolation [33]. Network slices consist of VNFs, physical network functions (PNFs), value added services, network and cloud resources from dedicated or shared software and hardware in the RAN, transport and core networks, combining different technologies such as SDN and NFV. Recently, NS has gained popularity among a large group of researchers from both academia and industry. Additionally, different standardization bodies such as 3GPP (3rd Generation Partnership Project), IETF (Internet Engineering Task Force) and ITU-T (International Telecommuni-

cation Union - Telecommunication Standardization Sector) are putting in a lot of effort in NS [4].

SDN and NFV/CNF complement each other towards building an end-to-end software defined and programmable network architecture. In LTE mobile core network, the data and control plane functions are realized by dedicated hardware that implements each specialized function. However, the 5G core is designed to be "cloud-native", in the sense that the functions that handle the control and data planes, e.g., User Plane Function (UPF) and Access and Mobility Management Function (AMF), could be deployed as VNFs/containers on a cloud infrastructure. NFV and SDN have been employed in networking middleboxes in the core and extended to RAN functions. The development of open and intelligent RAN (O-RAN) has received great attention [88]. O-RAN is proposed to enhance the RAN performance through virtualized network elements and open interfaces that incorporate intelligence into the RAN. O-RAN introduces programmable components that can run optimization routines with closed-loop control and orchestrate the RAN. Specifically, the O-RAN has logical controllers that monitor the status of the network (e.g., number of users, load, throughput, resource utilization) and process this data leveraging AI/ML algorithms to determine and apply control policies and actions in the RAN, for example, network and RAN slicing, load balancing, handovers and scheduling [14].

2.2 Machine Learning for Network Management and Control

Artificial Intelligence (AI) and Machine Learning (ML) play a vital role in facilitating closed-loop control and management, to realize self-driving automated network life-cycle management [9]. The recent success of ML techniques has driven the adoption of different learning perspectives to solving various networking problems. In this section, we present the three different ML settings: supervised ML, unsupervised ML, and reinforcement learning (RL). Deep learning [62] is a subset of machine learning that resembles biological nervous systems and performs representation learning via multi-layer transformations, that span over all three of the aforementioned learning paradigms. As deep learning has

a growing number of applications in mobile and wireless networking [135], we discuss the deep learning methods that are widely used in this domain.

2.2.1 Supervised Learning

Supervised learning consists of input variable x and an output variable $y = f(x)$. An algorithm is used to learn the mapping function f from x to y, so we can predict the output variable y for any new input data x. Supervised learning can be divided into regression and classification based on the continuity of the output [34]. Supervised learning is a very broad domain and has several learning algorithms, each with their own specifications and applications. In the following, the most common algorithms applied in the context of communication networks are presented.

- Support Vector Machine (SVM): SVM can be used for classification, regression and outliers detection, however, it is primarily used for classification problems. The idea behind an SVM classifier is to create the best boundary line, i.e. hyperplane that can separate n-dimensional space into distinct classes [127]. Linear SVM (as shown in Figure 2.2) is used for linearly separated data while non-linear SVM is used when the data cannot be classified by using a straight line. The SVM algorithm finds the nearest data point of the lines from both classes, i.e. support vectors and tries to maximize the distance between the support vector and the hyperplane (i.e. margin).

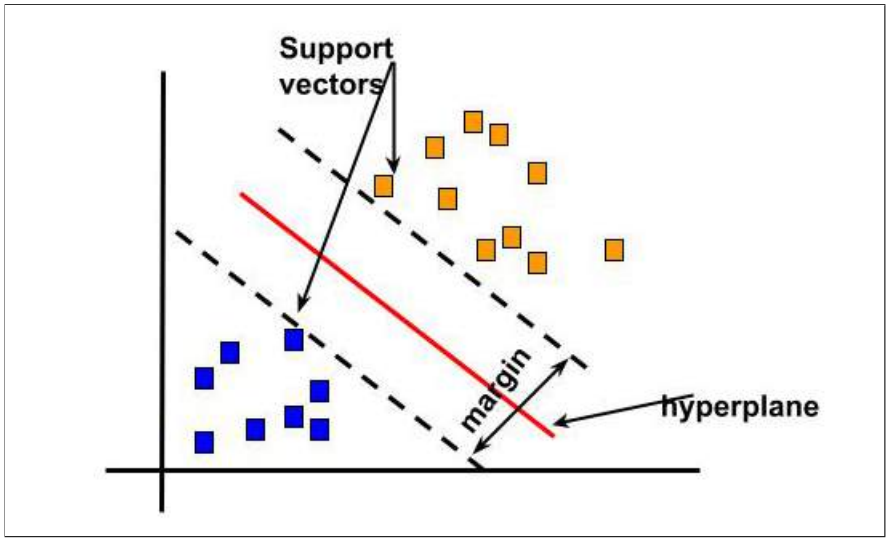


Figure 2.2: An illustration of Linear Support Vector machine (SVM).

- Decision Trees (DT): DT is powerful supervised method for both classification and regression. The basic idea of decision trees is splitting the original data into subsets based on an attribute value test as shown in Figure 2.3. This technique is repeated recursively on each derived subset, resulting in recursive partitioning. When the subset at a node has the same value of the target variable, or when splitting no longer adds value to the predictions, the splitting is complete [74]. Many supervised learning methods, such as Random Forests [34], Bagging [18], and Boosting [105] are built on the basis of DT.

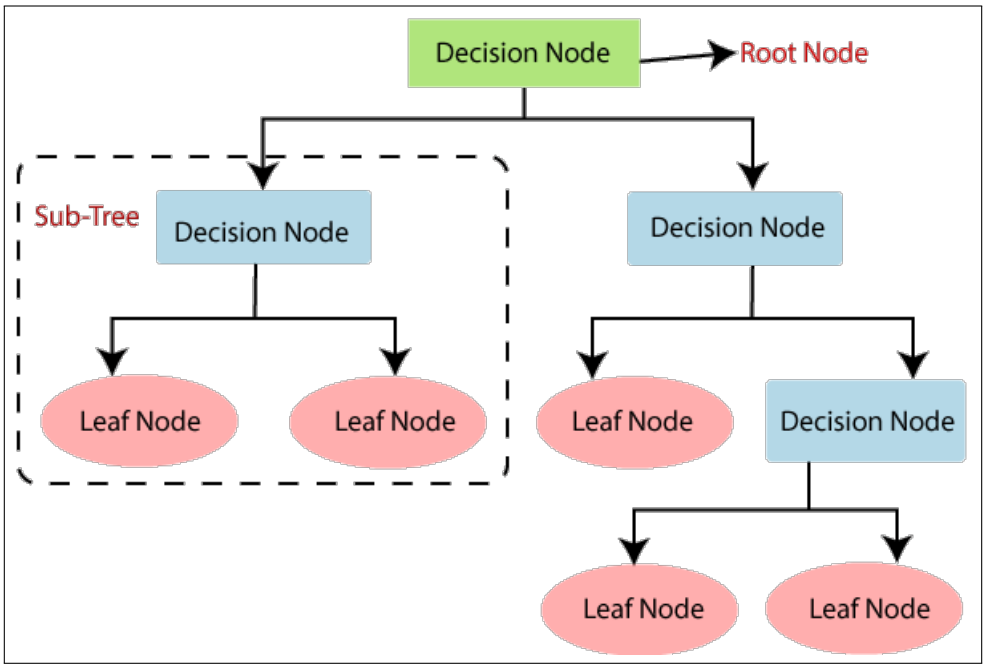


Figure 2.3: An illustration of Decision Trees (DT).

- Random Forest (RF): RF is an ensemble-based learning algorithm, more specifically is a version of the bagging method. It uses uncorrelated forest of decision trees to perform either classification or regression [34]. RF combines decision trees, each tree is comprised of various samples, called the bootstrap sample and a random subset of features. Instead of using the output of one decision tree, the final output of RF is generated using the majority vote for all trees in classification and the average in case of regression as shown in Figure 2.4. RF solves the problem of overfitting that may result from DT that uses all data in one tree. However, DT is more easier to interpret compared with RF.

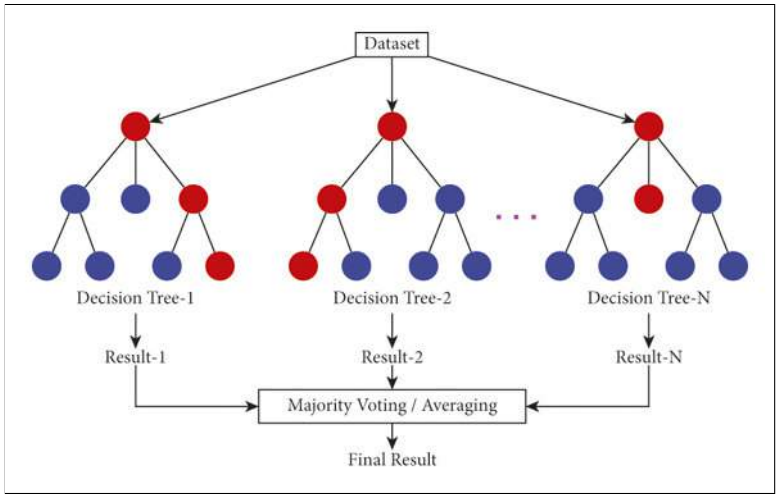


Figure 2.4: An illustration of Random Forest (RF) [55].

- Artificial Neural Networks (ANNs): ANN is a biologically-inspired computer structure that mimics the way biological neurons communicate with each other, where certain neurons (units) are activated given the current input, influencing the output of the neural network model. Figure. 2.5 depicts an example of a typical ANN which comprises of an input layer, one hidden layer, and an output layer. Each node connects to another and has a weight and threshold. If a node's output exceeds a certain threshold value, the node is activated, and data is sent to the next layer of the network. Otherwise, no data is sent to the network's next layer. With deep learning, ANNs can have many layers of linear and nonlinear transformations to form deep neural networks (DNNs) that can be used to model complex data representations. ANNs can be used for either classification (e.g., images classification) or regression (e.g., stock prices forecasting). ANN is trained via what is called "back-propagation" that is the process of fine-tuning the weights of ANN based on the error rate (i.e. loss) obtained in the previous epoch (i.e. iteration) aiming to have the weights values result in lower loss. Afterwards, LeCun and Bengio et al. [61] proposed the now popular Convolutional Neural Network (CNN) architecture , but development was stymied by the high computational cost of ANNs training and the limited processing capabilities of existing computers at the time. Later, Graphics processing units (GPUs) are developed to speed up the graphics processing and gradually adopted by deep learning community to accelerate the training processes of DNNs [104]. Thus, CNNs have been widely used for image recognition and computer vision to identify patterns within an image [59]. Subsequently, various

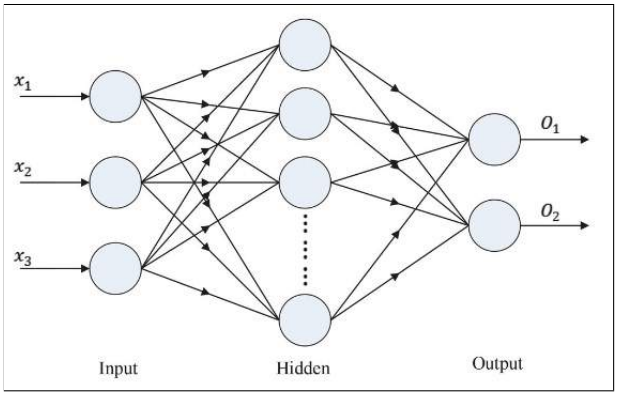


Figure 2.5: Architecture of typical Artificial Neural Network (ANN).

types of DNNs have been proposed for different purposes and applications such as Long Short-Term Memory networks (LSTM) [47], and Recurrent Neural Networks (RNNs) [80].

In the network management context, supervised learning is widely proposed in different problems, for example, congestion control, network resource allocation and management, traffic scheduling, performance prediction and load balancing [101]. Despite the superiority of supervised method, many related challenges are present. One of these problems is the data availability, mainly the labelled data for training supervised algorithms that is usually inaccessible due to privacy issues. Furthermore, compared to traditional ML models (e.g., SVM, DTs, and RF), deep learning methods achieve higher accuracy but less interpretation, making them unpreferable for network administrators.

2.2.2 Unsupervised Learning

Motivated by the above-mentioned challenges posed by supervised ML methods in solving wireless network problems, there is a surge of interest in the networking community to employ unsupervised ML methods to optimize network performance [123]. In unsupervised learning, the algorithm can explore the unlabeled data and discover some hidden patterns in the data. Some widely used unsupervised learning methods are given below.

- k-means clustering: It is the simplest unsupervised learning method used for classification. The idea behind K-means algorithm is to divide the unlabeled data into k different pre-defined clusters. Figure 2.6 demonstrates the k-means method; each cluster is associated with a centroid which is randomly selected. Then, the algorithm iteratively optimizes the positions of the centroids aiming to minimize the

sum of distances between the data points and their corresponding centroids [69]. It achieves a relatively good performance when there is a large distance between data points.

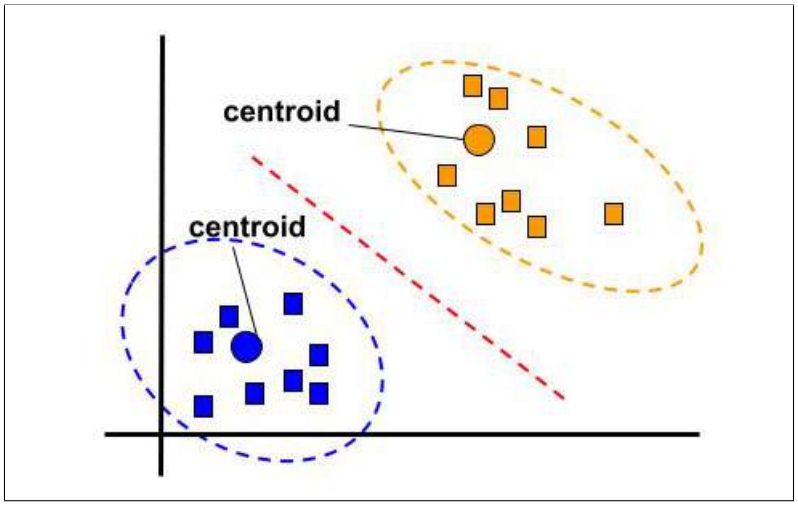


Figure 2.6: An illustration of k-means clustering method (k=2).

- Isolation Forest (iForest): iForest algorithm is based on decision trees to separate outliers from the rest of the data [71]. It is widely used for anomaly detection. Recursively, it partitions the data by randomly choosing a feature and then selecting a random split value, i.e. "cut-off-point" between the max and min values of that feature. The algorithm then determines if this isolates the data point; if so, it stops; otherwise, it selects a different feature and a different cut-off point at random. The anomalous data points will be distinguished from the rest of the data by this random splitting of features, which will result in shorter routes in trees (See Figure 2.7).

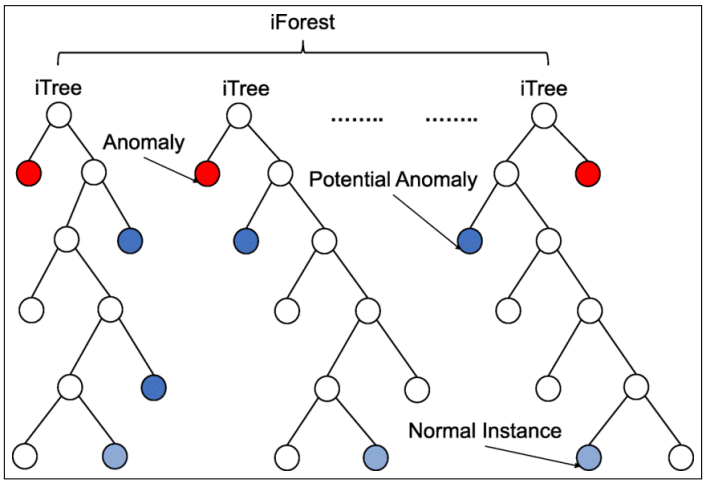


Figure 2.7: An illustration of Isolation Forest (iForest) [98].

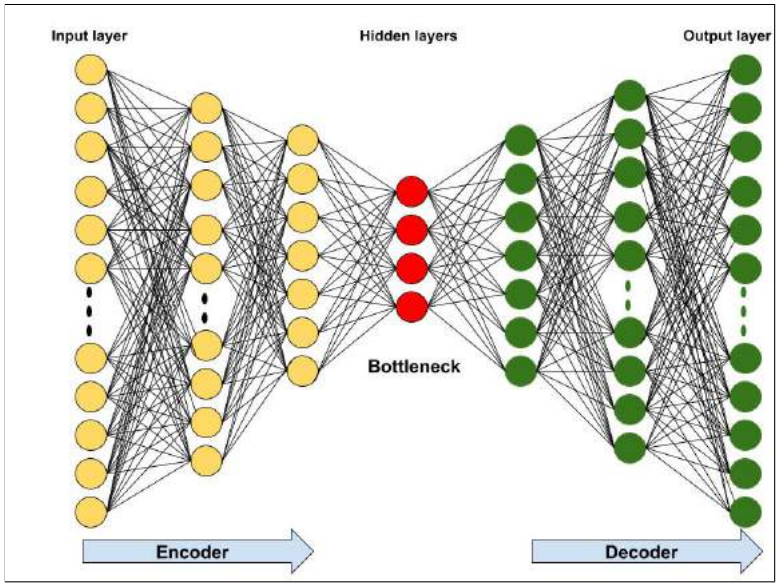


Figure 2.8: Architecture of Autoencoder (AE).

- Autoencoders (AEs): An autoencoder is a deep learning model based on ANNs used to learn latent variables of the input data in an unsupervised manner [103]. Figure 2.8 shows the architecture of autoencoder that comprises of encoder, bottleneck and decoder. The encoder maps the input data into a set of latent variables i.e. bottleneck , whereas the decoder maps the latent variables back into the input space as a reconstruction. An autoencoder learns to compress the data while minimizing the reconstruction error which represents the difference between the input data and the resulting reconstruction. Further, various extended versions from AEs are proposed such as Denoising Autoencoder (DAE) [125] and Variational Autoencoders (VAEs) [58]. AEs are widely used for dimensionality reduction and anomaly detection.

- Hierarchical Temporal Memory (HTM): HTM is a brain-inspired neural network model that mimics the structural properties of neocortex and the way the human brain processes information [44]. HTM employs online learning to automatically discover relationships between features in real-time streams to provide unsupervised predictions. Figure 2.9 shows the end-to-end framework for the HTM-based prediction system. First, the data stream is taken as input and encoded into a sparse distributed representation (SDR). An SDR is comprised of a large array of bits, i.e. zeros and ones. Each bit carries some semantic meaning that activates some bits (becomes 1's) where at any point in time most of the bits are 0's and the

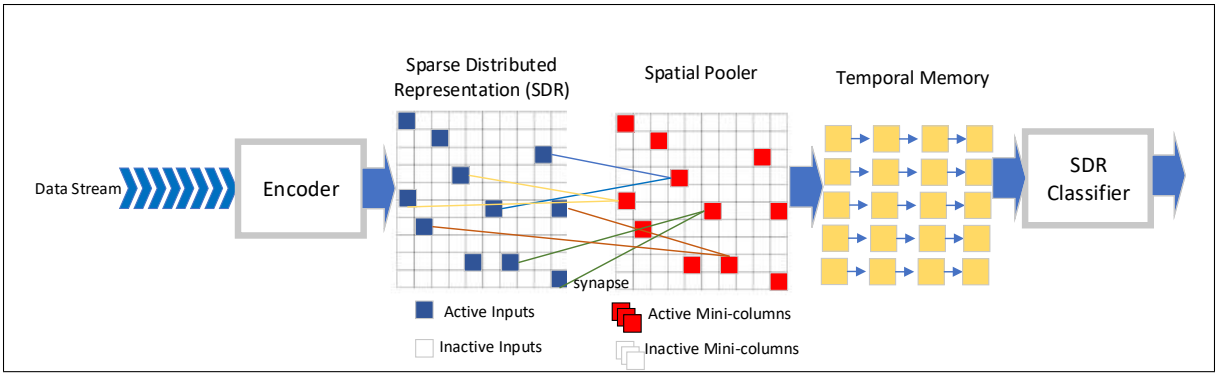


Figure 2.9: High-level architecture of the HTM system.

rest are 1's [95]. Next, the SDR is passed through the spatial pooler. The spatial pooler is responsible for feed-forward learning and data representations into columns and mini-columns. It models the activation of mini-columns given the feed-forward inputs [24]. The spatial pooler's output is fed into the temporal memory, which then outputs a prediction for the next activation. Learning in the spatial pooler and temporal memory is based on the connections, or synapses, between cells. The temporal memory learns connections between cells in the same layer while spatial pooler learns feed-forward connections between input SDRs and columns. Finally, the SDR classifier learns associations between the temporal memory predictions at current time and the predictions in the future. HTM has shown its efficiency and real-time capability of sequence prediction and anomaly detection [5]. It can learn both the temporal and spatial patterns using online training.

2.2.3 Reinforcement Learning

Reinforcement learning (RL) [117] is one of the most important machine learning paradigm, with a substantial influence on tackling a variety of real-world problems. RL is essentially a trial and error learning process in which an agent can take actions, receive a reward for taking those actions, and then adjust its strategy until the reward saturates i.e., optimal policy. However, in complex real-world problems, finding the optimal policy through trial and error learning consumes a lot of time to converge because the agent has to explore all possible actions in the environment. As a result, reinforcement learning's uses in practice are quite restricted. To overcome RL limitations and with aid of deep learning, researchers have proposed Deep Reinforcement Learning (DRL) that leverages the DNNs in the learning process to accelerate the convergence time. As a result, various applica-

tions have employed DRL instead of RL such as industry automation, computer vision, gaming and robotics [68]. Further, DRL has lately been employed as an emerging technology in the domains of networking and communications to efficiently tackle different problems.

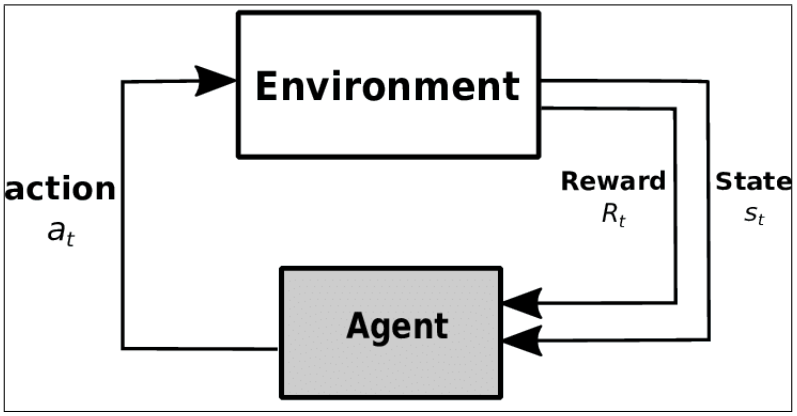


Figure 2.10: An illustration of Reinforcement Learning (RL).

The standard RL setup is depicted in Figure 2.10. It consists of an "Agent" that interacts with what is called "Environment" at each step of a discrete time series $t = 0, 1, 2, 3 \quad , T$. At each time step t the agent receives some representation of the Environment's state $s_t \in S$, where S is the set of possible states, and selects some action $a_t \in A$ to perform, where A is the set of possible actions. Reinforcement learning is mathematically formulated as a Markov Decision Process (MDP) [96] which is a discrete time stochastic control process. Typically, an MDP is defined by a tuple (S, A, P, r) where S is a finite set of states, A is a finite set of actions, P is a transition probability from state s to state $s(t+1)$ after action a_t is executed, and r_t is the is the immediate reward obtained after action a_t is performed. We denote π as a "*policy*" which is a mapping from a state to an action. A policy or a strategy describes the agent's behaviour at every time step t. The goal of an MDP is to find an optimal policy to maximize the reward function. An MDP can be finite or infinite time horizon. For the finite time horizon MDP, an optimal policy π^* to maximize the expected total reward is defined by $max_\pi \mathbb{E}[\sum_{t=0}^{T} r_t(s_t, \pi(s_t))]$, where $a_t = \pi(s_t)$. If the time horizon is infinite, the objective can be to maximize the expected discounted total reward by $lim_{T\to\infty} max_\pi \mathbb{E}[\sum_{t=0}^{T} \gamma r_t(s_t, \pi(s_t))]$, where $\gamma \in [0, 1]$ is the discount factor. A deterministic policy returns actions to be taken in each perceived state. On the other hand, a stochastic policy outputs a distribution over actions. Under a given policy π, we can define a value function or a Q-function which measures the expected

accumulated rewards staring from any given state s_t or any pair (s_t, a_t) and following the policy π, as shown below:

$$V^{\pi}(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | a_t \sim \pi(\ |s_t), s_0 = s] \tag{2.1}$$

$$Q^{\pi}(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | a_t \sim \pi(\ |s_t), s_0 = s] \tag{2.2}$$

To solve MDP, dynamic programming methods have been widely used, i.e. iterative algorithms (value iteration, Q-value iteration and policy iteration) [117]. However, such types of algorithms require that the model (environment transitions probability and rewards) is explicitly known which is not applicable for all RL problems. This drives the model-free RL algorithms that estimate the optimal policy without knowing the environment dynamics. In solving networking problems, several model-free RL methods have been used [77]. Below we discuss some of these methods by dividing them into three groups based on the agent's learned objective.

(a) Q-learning algorithms or value-based methods [130] learn the value of choosing an action on the current state (see Equ. 2.1) and the Q-value (see Equ. 2.2) by defining state-action pair when following policy π to maximizes the future reward. The Q-table represents the maximum future reward, for each state-action pair. In Q-learning algorithm, the agent can efficiently find an optimal policy when the state space and action space are discrete and small. However, when the state space or/and action space are large, the Q-table is not scalable and hence the Q-learning algorithm performance degrades. Thus, instead of using Q-table, DNNs is used to retrieve the state and calculate different Q-values for each action and then, the action with the highest Q-value is chosen, this is called Deep Q-Learning (DQL). Deep Q-Network (DQN) [85] algorithm is the initial algorithm of DQL that being proposed by DeepMind to play Atari video games. In DQN training, instead of using the recent actions, random samples from the replay memory that stores all experiences are used. Another DQL algorithm is Dueling DQN [129]; which defines advantage A of taking action a in state s besides the value V. Dueling DQN estimates both the value function and the action advantage function.

(b) Policy gradient (PG) methods [118] or policy-based methods search for the optimal agent's policy $\pi_\theta(a|s)$ by maximizing the agent's cumulative long-term reward J as in Equ. 2.3 with gradient ascent on the policy parameters θ^*. The policy gradients can be interpreted as in Equ. 2.4 and estimated using trajectories collected under the current policy. For each gradient update, the agent needs to interact with the environment and collect transitions.

$$J(\theta) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)] \tag{2.3}$$

$$\nabla_\theta J(\theta) = \mathbb{E}[\sum_{t=0}^{T} \nabla log \pi_\theta(a_t|s_t) Q^{\pi\theta}(s_t, a_t)] \tag{2.4}$$

In Equ. 2.4, $Q^{\pi\theta}$ is not known and needs to be estimated. Several approaches are possible such as Reinforce algorithm [131] which uses Monte Carlo to calculate the rewards as $\sum_{k=t}^{T} R(s_k, a_k)$. Additionally, Mnih et al. [84] presented Asynchronous Advantage Actor Critic (A3C); an actor-critic method that uses an actor to select the action while the critic estimates the value of being in a specific state. Then, the critic updates the actor with the optimal value function. A3C makes use of multiple agents, each of which has a replica of the environment and its own network parameters. All agents learn asynchronously with each iteration and contribute to the knowledge of the global network. The agent evaluates the rewards based on the value of the advantage $A = Q^\pi(s, a) - V^\pi(s)$. Further, various policy-based methods have been proposed such as Trust Region Policy Optimization (TRPO) algorithm [106] and Proximal Policy Optimization (PPO) algorithm [107].

(c) Deterministic Policy Gradient (DPG) algorithms are new class of algorithms that deal with continuous action space, first proposed by Silver et al. in [112]. The goal of the algorithm to maximize the long-term reward J in Equ. 2.3. Unlike a stochastic policy, deterministic policy $\mu_\theta(s)$ relays on the gradient of the state distribution ρ resulting in cumulative reward denoted by:

$$J(\mu_\theta) = \int_S \rho^\mu(s) \int_A \mu_\theta(s, a) r(s, a) da ds \tag{2.5}$$

Similar to Q-learning, for continuous actions the policy will move towards the maximum Q-value using the below gradient expectation:

$$\mathbb{E}_{s\sim\rho^{\mu^k}}[\nabla_\theta \mu_\theta(s) \nabla_a Q^{\mu^k}(s,a)|_{a=\mu_\theta(s)}] \tag{2.6}$$

Merging Equ. 2.5 and Equ. 2.6 will result in the below form of deterministic policy gradient:

$$\nabla_\theta J(\mu_\theta) = \int_S \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\pi(s,a)|_{a=\mu_\theta(s)} ds \tag{2.7}$$

Based on the deterministic policy gradient, different algorithms have been presented such as Deep Deterministic Policy Gradient (DDPG) [70]. DDPG is using actor-critic method to lean the Q-value and the deterministic policy $\mu_\theta(s)$ using Bellman equation. Specifically, DDPG algorithm maintains a parameterized actor function to specify the current policy by deterministically mapping states to a specific action while the critic is learned using the Bellman equation. Further, other DPG algorithms are proposed such as Twin Delayed Deep Deterministic Policy Gradient algorithm (TD3) [35].

Multi-Agent Reinforcement Learning (MARL). MARL involves using several agents at the same time. Similar to single agent RL which can be modeled as an MDP, MARL problem can be formulated as Markov game (MG) that is a dynamic game with probabilistic transitions played by multiple players, i.e. agents [110]. The simplest approach in multi-agent settings is to use agents that learn and act independent of each other, however, this approach is not applicable in most of the real applications. This is because each agent's policy changes during training, resulting in a non-stationary environment. In other words, a policy changed by an agent will influence the policy of the other agents and hence the lack of coordination will lead to conflicting policies. The training of multiple agents has long been a computational challenge. Since the complexity in the state and action space grows exponentially with the number of agents, even modern deep learning approaches may reach their limits [39]. If the training of agents is applied in a centralized manner, all information such as actions, observations and rewards from all gents should be sent to a centralized unit. In contrast to the centralized scheme, Lowe et al. [75] adopted

a training scheme called centralized training and decentralized execution in their proposal of Multi-Agent DDPG (MADDPG) algorithm. This approach assumes the existence of a centralized controller that collects extra information about the agents to ease training but not used during the normal operation of the system.

Various networking control problems can be modeled as MDPs or MGs, where DRL can be promising to solve these problems under the complex and dynamic networks environment. However, different challenges still exist. First, DRL methods need huge amount for data for training and testing which is not easily available, therefore most of the applications of DRL in networking are based on simulations that come with their own limitations. Second, most of the current DRL methods are assuming static environment which is not the case in the current heterogeneous networks. For example, 5G and beyond networks are supporting diverse applications and devices that generate different traffic patterns ending in highly dynamic environment. Moreover, this diversity increases the state space and consequently the agent needs longer training time to converge. Last but not least, most of networking applications are delay-sensitive and the network status is changing frequently, therefore the time needed to collect observations and execute actions should be kept minimal according to the addressed problem.

Chapter 3

Related Work and Research Contributions

In this chapter, we review the related work considering comprehensive surveys and recent publications in main journals and conferences. This review has been performed according to the three perspectives and aspects as defined by the thesis scope introduced in Section 1.2: anomaly detection in communication networks presented in Section 3.1, network management and control using DRL, presented in Section 3.2, and, finally AI-based data analytics in telemetry for network troubleshooting, presented in Section 3.3. We conclude each perspective with our contributions of this thesis.

3.1 Anomaly Detection in Communication Networks

In this section, we divide the related work into two parts: 1) time series anomaly detection algorithms or frameworks in general, and 2) network management related anomaly detection methods. We discuss them separately in the following and present our contributions related to them.

3.1.1 Time series Anomaly Detection Methods

Anomaly detection in time series has attracted much attention from the research community over the years. In this section, we list some of the recent works in anomaly detection in time series by categorizing them into supervised and unsupervised.

In traditional supervised anomaly detection, a binary classifier is trained using both normal and anomalous data; resulting in an imbalanced dataset without taking the temporal pattern of the data into account. Thus, for time series anomaly detection, the algorithm should be able to handle the temporal information in the data. Based on the nearest neighbor algorithm, Bagnall et al. [10] presented a novel classifier called NN-DTW which utilizes dynamic time warping similarity measure to achieve high accuracy. Consequently, various supervised classifiers have been presented as ensemble models based on nearest neighbor algorithm such as Proximity Forest [76]. Aligned with deep learning advances, Fawaz et al. [29] proposed InceptionTime which is a supervised time series classification method that consists of an ensemble of CNN models. Despite their superior performance in anomaly detection, supervised approaches are less prevalent than unsupervised methods due to the scarcity of labeled training data.

In contrast, various unsupervised methods have been presented for anomaly detection such as isolation methods that focus on separating outliers from the rest of the data. For instance, Isolation Forest (IF) [71] is an ensemble of binary decision trees; works in a way that the data samples that reside in trees branches are anomalies where as the samples that are far down in the tree are likely to be non-anomalous. Unsupervised approaches based on deep learning techniques, particularly autoencoder (AE) [103] based methods, have recently gained a lot of attention. These methods use the reconstruction error as the anomaly score, i.e. samples with a high score are considered to be anomalous. Zong et al. [138] proposed a method that uses a deep autoencoder and a Gaussian Mixture Model (GMM) to exploit the density distribution of multidimensional data without considering the temporal dependencies. Also, Park et al. [92] presented a variational autoencoder(VAE) model with LSTM; the LSTM-based encoder converts the input data and its temporal dependencies into a latent space. During decoding, it uses the latent space representation to estimate the output distribution. LSTM-VAE defines a sample as an anomaly when the log-likelihood of the current sample is below a threshold. Furthermore, Su et al. [114] demonstrated the effectiveness of using RNN to model the temporal dependencies between multivariate time series. They employed VAE for representation learning to map observations to stochastic variables and use the reconstruction probabilities of input samples as anomaly scores. Besides the autoencoder-based methods, different proposals have been presented which leverage the Generative Adversarial

Networks (GANs) [37] in detecting anomalies. For example, Li et al. [65] proposed a GAN framework to detecting anomalies in time series. The GAN network is combined with LSTMs to capture the temporal correlation of time series distributions. Moreover, Audibert et al. [7] presented USAD which combines the advantages of autoencoders and GANs. The autoencoder model is adversarially trained to amplify the reconstruction errors for anomalies so that the USAD model can detect the anomalies with small deviations from normal data. Feng et al. [31] introduced NSIBF, an unsupervised method for detecting operational failures in cyber-physical systems (CPS), which differs from earlier approaches. A neural network is employed to capture the dynamics of CPS using a state-space model which is aimed to address the instability of CPS data. A Bayesian filtering method is applied on the "identified" state-space model, which detects anomalies by estimating the likelihood of observed measurements over time.

3.1.2 Anomaly Detection Methods for Mobile Network Performance

In the context of mobile networks, anomaly is defined when the network experiences unexpected traffic pattern that is different than the usual behavior of the network. Mobile network performance is usually monitored using key performance indicators (KPIs). Several methods have been presented focusing on predicting anomalies in network traffic. For example, Hadj-Kacem et al. [41] proposed a proactive anomaly detection model that captures the correlation between the different KPIs using functional principal component analysis (FPCA). Then, they used the logistic regression classifier for the functional data to predict anomalies. Their model was evaluated on LTE dataset consists of four KPIs, namely average latency, average active user, downlink traffic volume and downlink load. They defined the anomaly if the value of one of the four KPIs exceeds a certain threshold. The logistic regression model achieved accuracy and F1-score of 71% and 70%, respectively. Also, Khatouni et al. [56] leveraged supervised classification models, namely SVM, DT, and LR to predict the latency in mobile broadband networks. They considered a large-scale dataset from three commercial mobile operators. However, they revealed that the use of traditional machine learning models to predict RTT in mobile broadband networks does not show high accuracy. Their results showed a performance of 71% (F1-score) when using DT. Alternatively, Sundqvist et al. [115] proposed an unsupervised model for

detecting latency anomalies in RAN. They used statistical, probabilistic, and clustering methods to automatically learn the functional behavior of RAN and identify where latency exceeds $1ms$ which resulted in an F1-score of 60%. Further, Ping et al. [94] formulated the cell outage detection problem in mobile networks as an anomaly detection problem. They proposed an autoencoder-based unsupervised model leveraging the measurement report information from user equipment (UE) namely, RSRP and RSRQ values of the serving cell and the neighboring cells, and the radio link failure (RLF). Another work based on unsupervised learning is proposed by Elsayed et al. [102] which is a deep learning framework based on LSTM-autoencoder and One-class SVM (OC-SVM) to detect abnormal traffic data. The LSTM-autoencoder is trained to learn the normal traffic pattern and to learn the compressed representation of the input data and then feed it to an OC-SVM model. The hybrid model overcomes the shortcomings of the separate OC-SVM, which has low capability to operate with massive and high-dimensional datasets.

Contributions: Detecting anomalous behavior on a set of correlated time series signals has been an active research area in the machine learning community for a long time [21, 91]. The aforementioned anomaly detection related work can be categorized into two main classes based on the algorithm structure: 1) traditional Ml-based and 2) deep learning-based. The nature of the current communication networks poses some difficulties for developing an anomaly detection system. One big challenge faced by deep learning-based methods is their feasibility. Since many networking applications are delay-sensitive, it is essential to design a real-time anomaly detection system with less computation loads. Moreover, deep learning algorithms need a large amount of data to perform well. Therefore, the prior efforts [56, 102, 41] mostly focus on traditional ML methods coming with the cost of the performance accuracy. Another challenge in networking, a large amount of data arrives at a rapid rate. The sheer volume of the data makes it difficult to store it in its entirety, and it is even more difficult to operate on it in real time. Furthermore, most of the communication networks such as mobile networks are distributed infrastructure composed of variety of probes and sensors across the entire network. Most of the current anomaly detection methods aggregate all the data to a single point and process them together. Such an approach, however, may become difficult to realise or infeasible in the future due to the exponentially growing size and scale of the network. Hence, carrying out an efficient and distributed anomaly detection remains challenging. Finally,

network behavior changes rapidly over time and would require frequent retraining of a static model which is specifically challenging with deep learning-based approaches. To tackle these challenges with both traditional ML and deep leaning methods for detecting anomalies in communication networks we present three different contributions. In the first part, our **Article I** employs traditional ML methods to predict high delays in mobile broadband networks. To improve the model accuracy we build a supervised ensemble of classifiers to detect increases in delay, unlike prior works that relay on one classification algorithm. Our proposed model examines whether jumps in round trip times (RTTs) have a pattern that can be predicted beforehand. We leverage per second RTT measurements from hundreds of probes in two LTE cellular networks. Besides the RTT measurements, we collect connection metadata which includes radio and connectivity parameters namely, Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ), Received Signal Strength Indicator (RSSI) and Radio Access Technology (RAT). We are interested in investigating whether high delays, the top 10%, can be predicted based on historical RTT values and available high-level metadata about connection quality. We construct a parsimonious explainable model that provides an accuracy of 80% and does not appear to be specific to a particular mobile operator. Further, we investigate whether our model can be extended to 5G using a small dataset with extra 5G metadata, resulting in an accuracy of 88%. Our model indicates that RTTs are long-range correlated and shows that radio measurements of channel occupancy are accurate predictors of the onset of high delays. Finally, our model does not require extensive retraining but rather a modest retraining with a weekly cycle.

In the second part, our **Article IV** proposes a system that Network Operations Centres and Support Centres for smaller operators can use in failure situations to identify the crosslayer network outages in a global ISP network. We leverage supervised learning algorithm, namely SVM to assist in classifying different outages. Our choice is highly dependent on the small dataset we collect in which the deep learning methods can't provide good results. Our system is two-stage; in the first stage, we passively collect Layer1-Layer3 data and classify the Layer2 events to an F1-score of 0.99. By adding a second stage with active monitoring to collect UDP ping data we predict other types of root causes with a 0.66 F1-score. Our analysis interestingly shows that Bidirectional

Forwarding Detection (BFD) features, which are the easiest to collect, give the best results for outage classification.

In the third part aligned with the research roadmap that evolves from supervised learning to unsupervised, we investigate the use of unsupervised learning in our dataset. Recent works on multivariate anomaly detection based on unsupervised deep learning [114, 7, 31] show promising results on several time series tasks, however, their performance on mobile network data needs to be further studied. In addition, such unsupervised learning methods are less applicable in many real-world scenarios because they need normal data for training, process data in batches and require considerable training data. To overcome these issues from state-of-the-art, our **Article II** presents a novel distributed anomaly detection architecture, RCAD, that combines HTM with intelligent model exchanging between different parts of the system, i.e. agents, to maximize anomaly detection accuracy. Compared with the above-mentioned approaches in Section 3.1.1, RCAD can process data streams in real time, learn the patterns of data continuously without training and improve the overall system performance by sharing the knowledge between models participating in the system. To the best of our knowledge, RCAD is the first system that proposes model replacement as a viable strategy for boosting accuracy. We implement and evaluate RCAD on real world measurements from a commercial mobile network. RCAD achieves over 0.7 F-1 score significantly outperforming the state-of-the-art methods.

3.2 Deep Reinforcement Learning in Mobile Network Management and Control

The advances in DRL have led to outstanding success in various domains. DRL has been recently proposed for solving many wireless communication problems. Several surveys summarized these works. For example, Luong et al. [77] provided a comprehensive overview of DRL application in communications and networking such as dynamic network access, data rate control, wireless caching, data offloading, network security and connectivity preservation. Compared to [77] which focused on single agent problems, Feriani et al. [32] presented an overview of both single-agent and multi-agent reinforcement learning as key enabling technologies of future wireless networks. They highlighted the potential for applying cooperative multi-agent reinforcement learning to different domains

such as mobile edge computing (MEC), unmanned aerial vehicles (UAV) networks and massive MIMO. Other surveys reviewed the application of deep reinforcement learning algorithms in specific domains such as internet of things (IoT) [63], URLLC in 6G networks [111], vehicular networks in 6G [121] and mobile edge caching [136]. In general, the previous proposals can be divided into two groups based on the action space. The first applies deep Q-learning to problems with a discrete action space, while the second applies actor-critic methods to problems with a continuous action space. Further, almost all existing work revolves around executing a single principal action. For example, Nasir et al. [87] proposed a power allocation scheme based on deep Q-learning model (DQN). Each transmitter collects channel state information and QoS information from several neighbors and adapts its own transmit power accordingly. Also, Li et al. [66] used DQN method to tackle the resource allocation multi-user computation offloading in wireless MEC. Recently, Ren et al. [99] addressed the problem of dynamic resource allocation for MEC slicing system using DDPG algorithm. They formulated the resource allocation problem as an MDP in which, the wireless resources and computing resources are configured dynamically according to the requirements of different types of slices to maximize the network operator revenue. Furthermore, Mei et al. [81] presented a hierarchical framework based on integrating DDPG algorithm and double deep-Q-network algorithm to solve RAN slicing problem. Specifically, this framework consists of two controllers: an upper-level controller which adjusts the slice configuration to improve QoS performance at a coarse granularity and a lower-level controller that schedules network resources and power allocation to active UEs in each network slice at a fine granularity. Their results showed that the physical resource blocks (RBs) in RAN slices can be managed efficiently using the DDPG for RB allocation. However, in this method, if the number of slices is different from the number of slices during training, RB allocation to slices is impossible. Therefore, RB allocation independent of the number of slices was proposed in [73], where Liu et al. presented a method called DeepSlicing which tackles the problem of resource allocation in multi-slicing networks in two stages. The first stage allocates resources to users within a slice through DRL that learns the optimal policy in each network slice to maximize the overall utilities of users in the slice while satisfying the users' service level agreement (SLA). The second stage coordinates the resource allocation across the network slices.

Contributions: The above-mentioned work focused on tracking and optimizing a single metric like coverage [124], power management [87], throughput [73] and resource sharing [99, 81] at a time. It flows directly from this limited focus that current approaches resort to often applying a single control action, e.g. adjusting base station transmit power. However, a multi-slice network is by definition a multi-service network. Hence, tracking a single metric is bound to assure quality for a subset of the services that run on the network. A viable approach to realizing self-driving mobile networks must be able assure quality for all running slices according to their priority and SLAs. Achieving this requires choosing and mixing diverse control actions, e.g. simultaneously adjusting coverage and optimizing resource allocation. In our **Article III** we aim to bridge this gap by proposing a DRL-based approach to manage resource utilization and performance in the RAN. Our approach tracks and optimizes diverse use cases. To this end, it employs several different control actions. We propose ICRAN, a novel control framework based on DRL that is capable of maximizing resource utilization and minimizing SLA violations in a multi-slice RAN. ICRAN introduces a novel reward function and an action space with diverse actions which allows for optimizing multiple objectives collectively, namely antenna tilt, traffic load balancing, and resource allocation. We investigate two different control architectures for our framework; centralized and distributed control. In the centralized framework, we formulate the problem as a single-agent DRL whereas, the distributed problem is solved via a multi-agent DRL. Finally, we validate the performance of our proposed framework through extensive simulations using ns-3. The experiment results show that our method outperforms the state-of-the-art methods in radio resources management. Moreover, the advantage of our method persists under different networking conditions such as high congestion and radio failure. Our evaluations show that ICRAN converges quickly to strategies that help maximizing radio resource utilization and minimizing SLA violations for the entire network. ICRAN outperforms approaches that leverage DRL, implements adaptive priority-based resource management as well as those resorting to heuristics to react to network changes. The benefit from ICRAN spans regimes where the network is lightly loaded, running at its capacity, and is heavily loaded. For example, ICRAN utilizes 97% of available radio resources when the network is loaded at 200% that is 7% higher than the next best method. At the same time, it reduces the number of SLA violations

for slices with stringent requirements, that is achieved by the next best method, by up to a factor of three.

3.3 AI-based Data Analytic in Telemetry

Software Defined Networking (SDN), through a separation of network control and data planes, has made network management tasks more versatile. Centralized network telemetry is often used in traditional SDN architecture [52]. Flow monitoring tools such as SNMP [20], sFlow [128], and NetFlow [3] are used to collect network telemetry data such as link congestion and link delay from different networking components. Monitored data is then transferred to the network controller to automate and optimize network management functionality by using machine learning algorithms on the centrally accessible data. One of the main goals of network monitoring in self-driving networks is to identify faults in the network and resolve the issues in a timely manner with minimal overhead. Thus, advanced monitoring solutions have been proposed to identify any performance degradation in the network, while attempting to balance high detection rates with minimal monitoring overhead costs. z-TORCH [108] is an automated NFV orchestration solution utilizing ML techniques to enhance quality of decisions in Management and Orchestration (MANO) systems. It adapts monitoring load based on VNF profile time variations. CellScope [89] uses domain specific knowledge to apply Multi-task Learning (i.e. use of several models in parallel) on RAN data. They are able to perform RAN troubleshooting and mobile phone energy bug diagnosis more efficiently compared to contemporary solutions. The authors of [122] proposed self-tuning, adaptive monitoring mechanism that adjusts measurement granularity based on observed traffic dynamics.

In the context of mobile core networks, P4 switches have been used for real-time attack detection and mitigation [15], enhancing the User Plane Function (UPF) functionality [78] and ensuring Quality-of-Service (QoS) at the slice level [100]. LossSight [120] tackles the problem of packet loss when using In-band Network Telemetry (INT), through "Alternate Marking" the telemetry headers of each flow. They are able to correctly identify and locate packet loss events, even when telemetry information is lost alongside the packets carrying it. There are also approaches that do not rely on P4. The authors of [52] proposed an

extension to INT tuned for wireless multihop networks. Each node runs both a telemetry module and an agent that defines the optimal local traffic engineering policy.

To troubleshoot network with anomalies in performance, knowledge about the causes of anomalies is the necessary. Hireche et al. [46] presented an AI-powered trustworthy distributed Self-driving network framework. The framework responds to in-band telemetry data through P4 and is able to modify the P4 code based on detected events. Moulay et al. [86] proposed an unsupervised ML method, Troubleshoot Trees (TTrees), to classify the reason of an anomaly in cellular performance with minimal amount of data and quick training. For a set of Key Performance Indicators (KPIs), TTree first uses a ML method to separate instances that cannot be classified. Then it uses a clustering method to group these unclassified instances, that are considered anomalies, into separate groups. At this latter stage an expert should inspect the clusters and assign that meaning [40].

Contributions: A cloudified mobile network is expected to deliver multitude of services, on parallel slices, with reduced capital and operating expenses. In the context of 5G mobile systems, to ensure that Service Level Agreements (SLA) of a customized end-to-end sliced service are met, it not only needs to monitor resource usage and characteristics of data flows on the virtualized network components and interfaces of its cloud mobile network but also to track performance at its radio interfaces and user equipments (UEs). With millions of UEs, a central monitoring architecture is not feasible. Our Article V proposes a distributed telemetry framework in which UEs act as early warning sensors. Upon flagging an anomaly, the cloudified mobile network activates its machine learning model to attribute the cause of the anomaly. For root cause analysis we employ active, passive and in-band telemetry in our network. Our framework achieves an impressive performance of 85% F1 score in detecting anomalies caused by different bottlenecks, and an overall 89% F1 score in attributing these bottlenecks.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

Traditional network management approaches cannot cope with the ever-growing networking services, applications, and the number of connected devices. Thus, a radical shift in the network management is needed. To this end, this thesis investigates the advances of AI/ML in network management automation. We utilize different learning paradigms to realize the proposed architecture of "self-driving networks" which is highly tied with intelligence and learning. First, we empirically investigated whether RTTs in mobile broadband networks could be accurately predicted. Using measurement data from a large number of probes, we found that a supervised binary ensemble learning-based model can accurately predict delay classes 80% and 88% of the time for 4G and 5G, respectively. Furthermore, our findings show that the accuracy of the model varies across probes and is a function of the delay profile of the probe. The model is more accurate for probes with high-delay episodes that last longer. Our model is transferable to other contexts that require only minor adjustments. This thesis also presents a supervised learning framework that Network Operations Centres and Support Centres for smaller operators can use to troubleshoot crosslayer outages in ISP networks. Our proposed model can classify 99% of Layer2 problems and 70% of other problems correctly. This is a significant improvement when we observe that only 35% of the customer cases we studied received any Reason for Outage (RFO) response from the Customer Support Centre.

While supervised methods appear to be effective for binary classification and anomaly detection in networking applications, they are facing challenges such as the availability of

labelled data and the need for retraining. Thus, this thesis presents RCAD, a Real-time Collaborative Anomaly Detection framework in mobile networks based on the unsupervised HTM and model replacement. The HTM model predicts anomalies in real time by learning sequences in mobile network data streams. Model replacement allows network probes with poor performing models to benefit from other probes that have better performing models. We evaluate RCAD on a two-week long dataset from 10 probes that are part of a setup for measuring the performance of commercial mobile broadband networks in Norway. RCAD has demonstrated superior performance to recent deep learning based methods in terms of F1-score. RCAD with DRL-based model replacement has achieved a level of performance comparable to a centralized HTM.

Inspired by the remarkable achievements of deep reinforcement learning in solving complex control problems in highly dynamic environments such as mobile network, this thesis proposes ICRAN, a novel control framework for optimizing resources utilization while minimizing SLA violations in a multi-slice RAN. ICRAN comprises two DRL-based architectures: centralized ICRAN and distributed ICRAN. Through extensive simulations using ns-3, we have confirmed the substantial advantages granted by ICRAN over other slicing schemes and recent works in terms of resources utilization and QoS assurance. ICRAN is, to the best of our knowledge, the only framework that simultaneously addresses multiple RAN problems.

Finally, as a proof of concept, this thesis proposes a distributed approach for timely collecting telemetry, detecting and attributing issues in mobile networks. We have built a software defined virtualised testbed that resembles a cloudified mobile network and used it to assess our telemetry architecture. We we have introduced a set of performance bottlenecks both in the RAN and the core. Based on ML methods, our system achieved an impressive performance of 85% F1 score in detecting bottlenecks by the end-user and overall 89% F1 score in attributing the bottlenecks based on network measurement.

4.2 Future Work

Several important research issues remain to be addressed in the future. Some of which we discuss next.

First, the findings in Article I are encouraging and can help multipath transport protocols such as multipath TCP (MPTCP) or multipath QUIC to decide which interface to use for sending the next packet. Thus, this work can be implemented in a multipath scheduler to proactively bound delays. In our analysis, we highlight that our model struggles when predicting short delay episodes. Therefore, it is important to investigate the means to improve the detection of short-lasting delay episodes such as more fine-grained measurements. Another step further in this research would be considering mobility cases in our measurements dataset.

Second, our model proposed in Article II has demonstrated superior performance to three deep learning-based methods namely OmniAnomaly [114], USAD [7] and NSIBF [31], however, this finding could only be made on mobile networks datasets. For future work, it would be interesting to test our method on applications with similar requirements as our networking use case in addition to looking deeper into different model exchange methods.

Third, even though our proposed intelligent control framework in Article III has succeeded in achieving its aim to optimize the overall network performance while satisfying the QoS requirements in multi-slice RAN, we highlight a number of limitations and enhancements that we plan to address in the future work. We need to include other types of traffic with different patterns such as IoT to support the increasing heterogeneous services and complex networks. Another issue is that our topology is relatively small, which may raise concerns about whether our framework can scale to much bigger networks. However, we believe that our preliminary results which have shown minimal differences between fully and partial observability distributed learning approaches are promising.

Last, our proposed telemetry architecture and bottleneck identification system presented in Article V has been implemented and evaluated in small testbed that comprises only one UE and eNB. We consider this as a limitation for our work and we plan to scale our testbed in the lab to include more UEs and assess our approach accordingly.

Bibliography

[1] IMT traffic estimates for the years 2020 to 2030. https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2370-2015-PDF-E.pdf. Date Accessed: Apr. 20, 2022.

[2] ns-3 Network Simulator. https://www.nsnam.org/. Date Accessed Nov. 10, 2021.

[3] Cisco IOS NetFlow. https://www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/index.html, July 2017. Date Accessed: May 12, 2022.

[4] Ibrahim Afolabi, Tarik Taleb, Konstantinos Samdanis, Adlen Ksentini, and Hannu Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys & Tutorials*, 20(3):2429–2453, 2018.

[5] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing*, 262:134–147, 2017.

[6] Azzam Alsudais and Eric Keller. Hey network, can you understand me? In *2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 193–198. IEEE, 2017.

[7] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3395–3404, 2020.

[8] AWS. Summary of the AWS Service Event in the Northern Virginia (US-EAST-1) Region. https://aws.amazon.com/message/12721/, December 2021. Date Accessed: Jan. 19, 2022.

[9] Sara Ayoubi, Noura Limam, Mohammad A Salahuddin, Nashid Shahriar, Raouf Boutaba, Felipe Estrada-Solano, and Oscar M Caicedo. Machine learning for cognitive network management. *IEEE Communications Magazine*, 56(1):158–165, 2018.

[10] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery*, 31(3):606–660, 2017.

[11] Ryan Beckett, Ratul Mahajan, Todd D. Millstein, Jitendra Padhye, and David Walker. Don't Mind the Gap: Bridging Network-wide Objectives and Device-level Configurations. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*, pages 328–341, 2016.

[12] Ryan Beckett, Ratul Mahajan, Todd D. Millstein, Jitendra Padhye, and David Walker. Network configuration synthesis with abstract topologies. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18-23, 2017*, pages 437–451, 2017.

[13] Roberto Bifulco and Gábor Rétvári. A survey on the programmable data plane: Abstractions, architectures, and open problems. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–7. IEEE, 2018.

[14] Leonardo Bonati, Salvatore D'Oro, Michele Polese, Stefano Basagni, and Tommaso Melodia. Intelligence and learning in O-RAN for data-driven NextG cellular networks. *IEEE Communications Magazine*, 59(10):21–27, 2021.

[15] Michel Bonfim, Marcelo Santos, Kelvin Dias, and Stenio Fernandes. A real-time attack defense framework for 5G network slicing. *Software: Practice and Experience*, 50(7):1228–1257, 2020.

[16] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review*, 44(3):87–95, 2014.

[17] Raouf Boutaba, Nashid Shahriar, Mohammad A Salahuddin, Shihabur R Chowdhury, Niloy Saha, and Alexander James. AI-driven Closed-loop Automation in 5G and beyond Mobile Networks. In *Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility*, pages 1–6, 2021.

[18] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[19] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

[20] Jeffrey D Case, Mark Fedor, Martin L Schoffstall, and James Davin. Simple network management protocol (SNMP), 1989.

[21] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.

[22] Xiaoqi Chen, Shir Landau Feibish, Yaron Koral, Jennifer Rexford, and Ori Rottenstreich. Catching the microburst culprits with snappy. In *Proceedings of the Afternoon Workshop on Self-Driving Networks*, pages 22–28, 2018.

[23] The P4 Language Consortium. Improving Network Monitoring and Management with Programmable Data Planes.

[24] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding. *Frontiers in computational neuroscience*, page 111, 2017.

[25] Santosh Janardhan, Engineering at Meta. Update about the October 4th outage. https://engineering.fb.com/2021/10/04/networking-traffic/outage/, October 2021. Date Accessed: Jan. 19, 2022.

[26] Ericsson. The cloud native transformation: A guide to cloud native design and operations principles. https://www.ericsson.com/assets/local/digital-services/offerings/core-network/5g-core-guide-cloud-native-transformation.pdf?_ga=2.39204577.

371240152.1614336546-1050635048.1612564305, October 2020. Date Accessed: Feb. 09, 2022.

[27] GR ETSI. Experiential Networked Intelligence (ENI); ENI use cases. 2018.

[28] ETSI NFV. Network functions virtualization: An introduction, benefits, enablers, challenges call for action, 2012. Darmstadt, Germany, SDN OpenFlow World Congr., White Paper.

[29] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. InceptionTime: Finding AlexNet for Time Series Classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020.

[30] Nick Feamster and Jennifer Rexford. Why (and How) Networks Should Run Themselves. In *Proceedings of the Applied Networking Research Workshop, ANRW 2018, Montreal, QC, Canada, July 16-16, 2018*, page 20, 2018.

[31] Cheng Feng and Pengwei Tian. Time series anomaly detection for cyber-physical systems via neural system identification and bayesian filtering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 2858–2867, 2021.

[32] Amal Feriani and Ekram Hossain. Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial. *IEEE Communications Surveys & Tutorials*, 2021.

[33] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K Marina. Network slicing in 5G Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100, 2017.

[34] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics New York, 2001.

[35] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[36] Piotr Gawłowicz and Anatolij Zubow. Ns-3 meets OpenAI gym: The playground for machine learning in networking research. In *Proceedings of the 22nd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 113–120, 2019.

[37] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[38] Ramesh Govindan, Ina Minei, Mahesh Kallahalla, Bikash Koley, and Amin Vahdat. Evolve or Die: High-Availability Design Principles Drawn from Googles Network Infrastructure. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*, pages 58–72, 2016.

[39] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2021.

[40] TPAW Group et al. In-band Network Telemetry (INT) data plane specification, June 2020.

[41] Imed Hadj-Kacem, Sana Ben Jemaa, Sylvain Allio, and Yosra Ben Slimen. Anomaly prediction in mobile networks: A data driven approach for machine learning algorithm selection. In *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*, pages 1–7. IEEE, 2020.

[42] Bo Han, Vijay Gopalakrishnan, Lusheng Ji, and Seungjoon Lee. Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, 53(2):90–97, 2015.

[43] Nikhil Handigol, Brandon Heller, Vimalkumar Jeyakumar, David Mazières, and Nick McKeown. I know what your packet did last hop: Using packet histories to troubleshoot networks. In *11th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 14)*, pages 71–85, 2014.

[44] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, page 23, 2016.

[45] Haiyang He, Yuanyuan Qiao, Sheng Gao, Jie Yang, and Jun Guo. Prediction of user mobility pattern on a network traffic analysis platform. In *Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, pages 39–44, 2015.

[46] Othmane Hireche, Chafika Benzaïd, and Tarik Taleb. Deep data plane programming and AI for zero-trust self-driven networking in beyond 5G. *Computer Networks*, 203:108668, 2022.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[48] P. Horn. Autonomic computing: IBM's perspective on the state of information technology. 2001.

[49] Mikio Iwamura. NGMN View on 5G Architecture. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[50] Arthur S Jacobs, Ricardo J Pfitscher, Rafael H Ribeiro, Ronaldo A Ferreira, Lisandro Z Granville, and Sanjay G Rao. Deploying Natural Language Intents with Lumi. In *Proceedings of the ACM SIGCOMM 2019 Conference Posters and Demos*, pages 82–84, 2019.

[51] Arthur Selle Jacobs, Ricardo José Pfitscher, Ronaldo Alves Ferreira, and Lisandro Zambenedetti Granville. Refining network intents for self-driving networks. *Computer Communication Review*, 48(5):55–63, 2018.

[52] Prabhu Janakaraj, Pinyarash Pinyoanuntapong, Pu Wang, and Minwoo Lee. Towards in-band telemetry for self driving wireless networks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 766–773. IEEE, 2020.

[53] Enio Kaljic, Almir Maric, Pamela Njemcevic, and Mesud Hadzialic. A survey on data plane flexibility and programmability in software-defined networking. *IEEE Access*, 7:47804–47840, 2019.

[54] Patrick Kalmbach, Johannes Zerwas, Péter Babarczi, Andreas Blenk, Wolfgang Kellerer, and Stefan Schmid. Empowering Self-Driving Networks. In *Proceedings*

of the Afternoon Workshop on Self-Driving Networks, SelfDN@SIGCOMM 2018, Budapest, Hungary, August 24, 2018, pages 8–14, 2018.

[55] Muhammad Yaseen Khan, Abdul Qayoom, Muhammad Suffian Nizami, Muhammad Shoaib Siddiqui, Shaukat Wasi, and Syed Muhammad Khaliq-ur-Rahman Raazi. Automated Prediction of Good Dictionary EXamples (GDEX): A Comprehensive Experiment with Distant Supervision, Machine Learning, and Word Embedding-Based Deep Learning Techniques. *Complexity*, 2021, 2021.

[56] Ali Safari Khatouni, Francesca Soro, and Danilo Giordano. A machine learning application for latency prediction in operational 4G networks. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 71–74. IEEE, 2019.

[57] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*, 2015.

[58] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[60] Amund Kvalbein, Džiugas Baltrūnas, Kristian Evensen, Jie Xiang, Ahmed Elmokashfi, and Simone Ferlin-Oliveira. The Nornet Edge Platform for Mobile Broadband Measurements. *Computer Networks*, 61:88–101, 2014.

[61] Yann LeCun, Yoshua Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[62] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[63] Lei Lei, Yue Tan, Kan Zheng, Shiwen Liu, Kuan Zhang, and Xuemin Shen. Deep reinforcement learning for autonomous internet of things: Model, applications and challenges. *IEEE Communications Surveys & Tutorials*, 22(3):1722–1760, 2020.

[64] Bing Li, Shengjie Zhao, Rongqing Zhang, Qingjiang Shi, and Kai Yang. Anomaly detection for cellular networks using big data analytics. *IET Communications*, 13(20):3351–3359, 2019.

[65] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*, pages 703–716. Springer, 2019.

[66] Ji Li, Hui Gao, Tiejun Lv, and Yueming Lu. Deep reinforcement learning based computation offloading and resource allocation for MEC. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.

[67] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. Flowradar: A better netflow for data centers. In *13th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 16)*, pages 311–324, 2016.

[68] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[69] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[70] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[71] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.

[72] Hongqiang Harry Liu, Xin Wu, Wei Zhou, Weiguo Chen, Tao Wang, Hui Xu, Lei Zhou, Qing Ma, and Ming Zhang. Automatic Life Cycle Management of Network Configurations. In *Proceedings of the Afternoon Workshop on Self-Driving Networks, SelfDN@SIGCOMM 2018, Budapest, Hungary, August 24, 2018*, pages 29–35, 2018.

[73] Qiang Liu, Tao Han, Ning Zhang, and Ye Wang. DeepSlicing: Deep reinforcement learning assisted resource allocation for network slicing. In *GLOBECOM 2020-2020 IEEE Global Communications Conference*, pages 1–6. IEEE, 2020.

[74] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[75] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[76] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Zaidi, Bart Goethals, François Petitjean, and Geoffrey I Webb. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, 33(3):607–635, 2019.

[77] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3133–3174, 2019.

[78] Robert MacDavid, Carmelo Cascone, Pingping Lin, Badhrinath Padmanabhan, Ajay Thakur, Larry Peterson, Jennifer Rexford, and Oguz Sunay. A P4-based 5G User Plane Function. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*, pages 162–168, 2021.

[79] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review*, 38(2):69–74, 2008.

[80] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.

[81] Jie Mei, Xianbin Wang, Kan Zheng, Gary Boudreau, Akram Bin Sediq, and Hatem Abou-zeid. Intelligent Radio Access Network Slicing for Service Provisioning in

6G: A Hierarchical Deep Reinforcement Learning Approach. *IEEE Transactions on Communications*, 2021.

[82] Oliver Michel and Eric Keller. SDN in wide-area networks: A survey. In *2017 Fourth International Conference on Software Defined Systems (SDS)*, pages 37–42. IEEE, 2017.

[83] MIT. Network automation: Efficiency, resilience, and the pathway to 5G. https://www.technologyreview.com/s/613533/network-automation-efficiency-resilience-and-the-pathway-to-5g/, April 2020. Accessed May 5, 2022.

[84] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[85] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[86] Mohamed Moulay, Rafael Garcia Leiva, Vincenzo Mancuso, Pablo J Rojo Maroni, and Antonio Fernandez 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)*, pages 199–208. IEEE, 2021.

[87] Yasar Sinan Nasir and Dongning Guo. Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks. *IEEE Journal on Selected Areas in Communications*, 37(10):2239–2250, 2019.

[88] O-RAN Alliance. O-RAN: Towards an Open and Smart RAN, White Paper, Oct. 2018.

[89] Anand Padmanabha Iyer, Li Erran Li, Mosharaf Chowdhury, and Ion Stoica. Mitigating the latency-accuracy trade-off in mobile data analytics systems. In *Pro-*

ceedings of the 24th Annual International Conference on Mobile Computing and Networking, pages 513–528, 2018.

[90] Tian Pan, Enge Song, Zizheng Bian, Xingchen Lin, Xiaoyu Peng, Jiao Zhang, Tao Huang, Bin Liu, and Yunjie Liu. Int-path: Towards optimal path planning for in-band network-wide telemetry. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 487–495. IEEE, 2019.

[91] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)*, 54(2):1–38, 2021.

[92] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters*, 3(3):1544–1551, 2018.

[93] Ben Pfaff, Justin Pettit, Teemu Koponen, Ethan Jackson, Andy Zhou, Jarno Rajahalme, Jesse Gross, Alex Wang, Joe Stringer, Pravin Shelar, et al. The design and implementation of open vswitch. In *12th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 15)*, pages 117–130, 2015.

[94] Yeh-Hong Ping and Po-Chiang Lin. Cell outage detection using deep convolutional autoencoder in mobile communication networks. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1557–1560. IEEE, 2020.

[95] Scott Purdy. Encoding data for HTM systems. *arXiv preprint arXiv:1602.05925*, 2016.

[96] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[97] Darijo Raca, Ahmed H Zahran, Cormac J Sreenan, Rakesh K Sinha, Emir Halepovic, Rittwik Jana, Vijay Gopalakrishnan, Balagangadhar Bathula, and Matteo Varvello. Empowering video players in cellular: Throughput prediction from radio network measurements. In *Proceedings of the 10th ACM Multimedia Systems Conference*, pages 201–212, 2019.

[98] Yousra Regaya, Fodil Fadli, and Abbes Amira. Point-Denoise: Unsupervised outlier detection for 3D point clouds enhancement. *Multimedia Tools and Applications*, 80(18):28161–28177, 2021.

[99] Yin Ren, Aihuang Guo, Chunlin Song, and Yidan Xing. Dynamic Resource Allocation Scheme and Deep Deterministic Policy Gradient-Based Mobile Edge Computing Slices System. *IEEE Access*, 2021.

[100] Ruben Ricart-Sanchez, Pedro Malagon, Antonio Matencio-Escolar, Jose M Alcaraz Calero, and Qi Wang. Toward hardware-accelerated QoS-aware 5G network slicing based on data plane programmability. *Transactions on Emerging Telecommunications Technologies*, 31(1):e3726, 2020.

[101] Mohammad Azmi Ridwan, Nurul Asyikin Mohamed Radzi, Fairuz Abdullah, and YE Jalil. Applications of machine learning in networking: a survey of current issues and future challenges. *IEEE Access*, 2021.

[102] Mahmoud Said Elsayed, Nhien-An Le-Khac, Soumyabrata Dev, and Anca Delia Jurcut. Network anomaly detection using LSTM based autoencoder. In *Proceedings of the 16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, pages 37–45, 2020.

[103] Mayu Sakurada and Takehisa Yairi. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*, pages 4–11, 2014.

[104] Jason Sanders and Edward Kandrot. *CUDA by example: an introduction to general-purpose GPU programming*. Addison-Wesley Professional, 2010.

[105] Robert E Schapire and Yoav Freund. Boosting: Foundations and algorithms. *Kybernetes*, 2013.

[106] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[107] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[108] Vincenzo Sciancalepore, Faqir Zarrar Yousaf, and Xavier Costa-Perez. z-TORCH: An automated NFV orchestration and monitoring solution. *IEEE Transactions on Network and Service Management*, 15(4):1292–1306, 2018.

[109] Muhammad Shahbaz, Sean Choi, Ben Pfaff, Changhoon Kim, Nick Feamster, Nick McKeown, and Jennifer Rexford. Pisces: A programmable, protocol-independent software switch. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 525–538, 2016.

[110] Lloyd S Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39(10):1095–1100, 1953.

[111] Changyang She, Chengjian Sun, Zhouyou Gu, Yonghui Li, Chenyang Yang, H Vincent Poor, and Branka Vucetic. A Tutorial on Ultra-reliable and low-latency communications in 6G: integrating domain knowledge into deep learning. *Proceedings of the IEEE*, 109(3):204–246, 2021.

[112] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[113] AG Spilling, AR Nix, MA Beach, and TJ Harrold. Self-organisation in future mobile communications. *Electronics & Communication Engineering Journal*, 12(3):133–147, 2000.

[114] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2828–2837, 2019.

[115] Tobias Sundqvist, Monowar Bhuyan, and Erik Elmroth. Uncovering latency anomalies in 5G RAN-A combination learner approach. In *2022 14th International Conference on COMmunication Systems & NETworkS (COMSNETS)*, pages 621–629. IEEE, 2022.

[116] Yu-Wei Eric Sung, Xiaozheng Tie, Starsky H. Y. Wong, and Hongyi Zeng. Robotron: Top-down Network Management at Facebook Scale. In *Proceedings of the ACM*

SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016, pages 426–439, 2016.

[117] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[118] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[119] Tarik Taleb, Adlen Ksentini, and Bruno Sericola. On service resilience in cloud-native 5G mobile systems. *IEEE Journal on Selected Areas in Communications*, 34(3):483–496, 2016.

[120] Lizhuang Tan, Wei Su, Wei Zhang, Huiling Shi, Jingying Miao, and Pilar Manzanares-Lopez. A packet loss monitoring system for in-band network telemetry: Detection, localization, diagnosis and recovery. *IEEE Transactions on Network and Service Management*, 18(4):4151–4168, 2021.

[121] Fengxiao Tang, Yuichi Kawamoto, Nei Kato, and Jiajia Liu. Future intelligent and secure vehicular network toward 6G: Machine-learning approaches. *Proceedings of the IEEE*, 108(2):292–307, 2019.

[122] Gioacchino Tangari, Daphne Tuncer, Marinos Charalambides, Yuanshunle Qi, and George Pavlou. Self-adaptive decentralized monitoring in software-defined networks. *IEEE Transactions on Network and Service Management*, 15(4):1277–1291, 2018.

[123] Muhammad Usama, Junaid Qadir, Aunn Raza, Hunain Arif, Kok-Lim Alvin Yau, Yehia Elkhatib, Amir Hussain, and Ala Al-Fuqaha. Unsupervised machine learning for networking: Techniques, applications and research challenges. *IEEE access*, 7:65579–65615, 2019.

[124] Filippo Vannella, Grigorios Iakovidis, Ezeddin Al Hakim, Erik Aumayr, and Saman Feghhi. Remote Electrical Tilt Optimization via Safe Reinforcement Learning. In *2021 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–7. IEEE, 2021.

[125] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[126] Jing Wang, Jian Tang, Zhiyuan Xu, Yanzhi Wang, Guoliang Xue, Xing Zhang, and Dejun Yang. Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[127] Lipo Wang. *Support vector machines: theory and applications*, volume 177. Springer Science & Business Media, 2005.

[128] Mea Wang, Baochun Li, and Zongpeng Li. sFlow: Towards resource-efficient and agile service federation in service overlay networks. In *24th International Conference on Distributed Computing Systems, 2004. Proceedings.*, pages 628–635. IEEE, 2004.

[129] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Hasselt, Marc Lanctot, and Nando Freitas. Dueling network architectures for deep reinforcement learning. In *International conference on machine learning*, pages 1995–2003. PMLR, 2016.

[130] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[131] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[132] Q Wu, J Strassner, A Farrel, and L Zhang. Network telemetry and big data analysis. *Network Working Group Internet-Draft*, 2016.

[133] Zhenjie Yang, Yong Cui, Baochun Li, Yadong Liu, and Yi Xu. Software-defined wide area network (SD-WAN): Architecture, advances and opportunities. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9. IEEE, 2019.

[134] Minlan Yu. Network telemetry: towards a top-down approach. *ACM SIGCOMM Computer Communication Review*, 49(1):11–17, 2019.

[135] Chaoyun Zhang, Paul Patras, and Hamed Haddadi. Deep learning in mobile and wireless networking: A survey. *IEEE Communications surveys & tutorials*, 21(3):2224–2287, 2019.

[136] Hao Zhu, Yang Cao, Wei Wang, Tao Jiang, and Shi Jin. Deep reinforcement learning for mobile edge caching: Review, new features, and open issues. *IEEE Network*, 32(6):50–57, 2018.

[137] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert Greenberg, Guohan Lu, Ratul Mahajan, Dave Maltz, Lihua Yuan, Ming Zhang, Ben Y Zhao, et al. Packet-level telemetry in large datacenter networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 479–491, 2015.

[138] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*, 2018.

Part II: Research Papers

Article I

Ahmed, A. H., Hicks, S., Riegler, M. A., and Elmokashfi, A. (2021). **Predicting High Delays in Mobile Broadband Networks.** IEEE Access, 9, 168999-169013.
DOI: **https://doi.org/10.1109/ACCESS.2021.3138695**.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 15, 2021, accepted December 23, 2021, date of publication December 24, 2021,
date of current version December 31, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3138695

Predicting High Delays in Mobile Broadband Networks

AZZA H. AHMED[1,2], (Member, IEEE), STEVEN HICKS[1,2], (Member, IEEE), MICHAEL ALEXANDER RIEGLER[1], AND AHMED ELMOKASHFI[1], (Member, IEEE)

[1]SimulaMet—Simula Metropolitan Center for Digital Engineering, 1325 Oslo, Norway
[2]Department of Computer Science, OsloMet—Oslo Metropolitan University, 0167 Oslo, Norway

Corresponding author: Azza H. Ahmed (azza@simula.no)

ABSTRACT The number of applications that run over mobile networks, expecting bounded end-to-end delay, is increasing steadily. However, the stochastic and shared nature of the wireless medium makes providing such guarantees challenging. Using several network interfaces simultaneously can help address fluctuating delays, provided that transport protocols can switch between them in a timely manner. Today's protocols are mostly closed-loop and thus require at least one round trip before reacting to increased delay. This paper examines whether jumps in round trip times (RTTs) have a pattern that can be predicted beforehand. Using per second RTT measurements from hundreds of probes in two Long Term Evolution (LTE) cellular networks, we train an ensemble of classifiers to detect increases in delay. We construct a parsimonious explainable model that provides an accuracy of 80% and does not appear to be specific to a particular mobile operator. Further, we examine whether our model can be extended to 5G using a small dataset with extra 5G metadata, resulting in an accuracy of 88%. Our model indicates that RTTs are long-range correlated and shows that radio measurements of channel occupancy are accurate predictors of the onset of high delays. These results suggest that it is feasible to build an open-loop control system for multiplexing among several interfaces to proactively bound delays.

INDEX TERMS Delay, prediction, machine learning, LTE, 5G.

I. INTRODUCTION

Guaranteeing low and stable end-to-end delay over mobile networks is one of the key motivations for 5G. Ultra-reliable low-latency communication is one of three use cases 5G is envisioned to cater for [1]. Reliable latency is important for supporting interactive applications such as hepatic control, virtual and augmented reality, and critical applications such as smart grid metering and public safety communication.

Delays can increase for a number of reasons, including interference, handover, and congestion both in the radio and beyond [2], [3]. New error correction mechanisms and novel radio access strategies, such as the flexible numerology introduced by 5G new radio [4], may help drive delay down [5]. However, addressing congestion and handover remains challenging because of the stochastic, shared and time-slotted nature of the wireless medium.

Leveraging the availability of several radios per end device has also emerged as a potential approach to bound performance unpredictability. Previous studies have shown that network availability can be boosted to five nines by connecting to two mobile operators simultaneously and the throughput can be enhanced markedly [6], [7]. Several multipath transport protocols, such as Multipath Transmission Control Protocol (MP-TCP) and QUIC multipath, are standardized to support the simultaneous use of multiple links [8], [9]. These protocols use a scheduler that monitors the state of each link in use before deciding which link to use next. Similar to TCP, performance monitoring is essentially a closed-loop that requires at least a single round trip, but often several, before taking a qualified decision. Unfortunately, this waiting time can be too long to meet the expectations of delay-sensitive applications.

To address these limitations, we ask the simple question of RTTs over mobile networks can be predicted by end devices. We are not interested in the exact value of the RTT, but rather whether it falls below or above a certain threshold. Furthermore, the prediction should be based only on measurements and metadata that are available to end devices, like for example, signal strength. Accurate predictions can help

The associate editor coordinating the review of this manuscript and approving it for publication was Ehab Elsayed Elattar.

TABLE 1. List of abbreviations.

Abbreviation	Description
RTT	Round Trip Time
RSRP	Reference Signal Received Power
RSRQ	Reference Signal Received Quality
RSSI	Received Signal Strength Indicator
RAT	Radio Access Technology
MCS	Modulation Coding Scheme
NSA	Non-Stand Alone
UE	User Equipment
NNE	NorNet Edge
ARIMA	AutoRegressive Integrated Moving Average
LR	Logistic Regression
RF	Random Forest
SVM	Support Vector Machine
DT	Decision Tree
ROC	Receiver Operating Characteristic
AUC	Area Under the Curve
MCC	Matthews Correlation Coefficient
SMOTE	Synthetic Minority Oversampling Technique

transport protocols to short-circuit the closed control loop by making local decisions instead of waiting for at least one RTT.

We leveraged end-to-end measurements and metadata from a large number of stationary probes connected to two mobile operators over LTE. Then, we trained a number of machine learning classifiers to verify whether delays could be reliably predicted. We focused on stationary measurements because it was the simplest scenario and thus succeeding in predicting stationary delays is the first step towards scenarios with complex mobility. Furthermore, many use cases with stringent delay requirements are associated with low to no mobility (e.g., smart meters). Interestingly, we found that a binary ensemble classifier could accurately predict low and high delay in 80% of the cases. In fact, the classifier also predicted correctly 75% of the worst 10% of the RTTs. More importantly, the model is interpretable and transferable to other network operators and requires minimal retraining to remain effective over an extended period. Moreover, we tested our classification model on a small 5G dataset of RTT measurements and extra metadata. The model achieved an accuracy of 88% for classifying the delays. Our findings can be readily used to improve the performance of multipath protocols when using several wireless links for bounding delays.

The rest of the paper is organized as follows: We present our measurement data in Sec. II. We then discuss our approach for predicting delays in Sec. III and present the prediction results in Sec. IV. Sections V and VI dig deeper into failed predictions and examine the prediction accuracy over time. In Sec. VII, we investigate the performance of our model on 5G data. We review related work in Sec. VIII. The main findings are discussed in Sec. IX before concluding in Sec. X

II. MEASUREMENT DATA

In this section, we describe our measurement setup, dataset, and pre-processing steps.

We studied RTT measurements from a set of geographically spread stationary probes. These probes are part of

FIGURE 1. Measurement node. The red box encloses the single board computer. The box also includes a smart power socket that can be rebooted via SMS.

the NorNet Edge (NNE) platform, which is a country-wide setup for measuring commercial mobile broadband networks in Norway. The probe is a single-board computer that runs Linux and connects to at least two mobile operators using commercial off-the-shelf user equipment (UE) and subscriptions. More specifically, we use the APU2 platform from PC Engines (see Figure 1).[1] Our board is equipped with a quad core CPU, 4GB RAM and two miniPCI slots. To connect to commercial mobile networks, we use the Sierra Wireless AirPrime MC7455 miniPCI modem, which supports LTE CAT 6 (LTE-advanced).[2] The modem uses external antenna, which are visible in Figure 1. To enhance the availability of the nodes, we attach them to a smart power socket that can be power-cycled remotely via SMS. Our probes conduct end-to-end measurements to a set of well provisioned servers that we control, these include delay, packet loss, and speed. Figure 2 illustrates the measurement setup. An NNE node connects to the Internet via commercial mobile subscriptions and performs end-to-end measurements to the NNE backend.

In this study, we consider measurements from the two largest mobile operators in Norway, which we refer to as Op_1 and Op_2 in the sequel. The probes measure RTTs by sending a 20-bytes User Datagram protocol (UDP) packet every second, over all available connections, to a well-provisioned server that echoes it back. We focused on the RTT measurements collected over LTE during September and October 2018. The Op_1 dataset includes more than 44.96 million RTT data points from 79 probes, while the Op_2 dataset includes approximately 14.47 million data points from 77 probes. The

[1] https://www.pcengines.ch/apu2.htm
[2] https://www.sierrawireless.com/iot-solutions/products/mc7455/

difference between the two datasets stems from the fact that Op_2 connections were on 3G for a non-trivial duration, and this data had to be filtered out. At the time of the study, Op_2 did not implement handover between radio access technologies while a UE was actively sending data, i.e. data sent by a UE over 3G would not be handed over to 4G. Therefore, many of the connections to Op_2 were on 3G for an extended period of time. Besides filtering out these periods, we removed all instances where a probe underwent maintenance or the NNE backend had issues. The NNE backend is connected to the Internet via a well provisioned link through a research and educational network. However, to avoid including times where the measured RTTs were influenced by congestion in the research and educational network, we filtered all measurements where a large fraction of probes, across operators, registered larger than usual RTTs.

In addition to the active measurements, the probes collect connection metadata. These include radio and connectivity parameters, which are listed below.

- **Received signal strength indicator (RSSI)** is a measure of the power received by the UE, including both the signal and noise.
- **Reference signal received power (RSRP)** is a measure of the power in the LTE reference signal and is averaged over the entire bandwidth. RSRP is a more accurate estimate of the received useful power.
- **Reference signal received quality (RSRQ)** is a measure of the quality of the received signal. A low RSRQ often coincides with a loaded cell.
- **Radio access technology (RAT)** indicates mobile generation in use, that is, 2G, 3G, and 4G.

The metadata is collected every minute, as well as whenever there is a change. We associated every RTT measurement with the closest past metadata value. Next, we removed all RTT measurements without the corresponding metadata. The removed fractions are 0.2% and 0.3% for Op_1 and Op_2, respectively. Finally, we checked the sanity of the metadata values and removed all RTT measurements that were associated with metadata values outside the correct value ranges, that is, RSRP (-44dBm to -140dBm), RSRQ (-3dB to -20dB), and RSSI (-6dBm to -100dBm).

III. PREDICTING ROUND TRIP DELAY

This section takes a closer look at our dataset and approaches to predict delays.

A. RTT MEASUREMENTS

Figure 3 shows the distribution of RTTs for Op_1 and Op_2. There were no clear differences between the two operators for the bulk of the initial part of the distributions. Approximately 60% of the RTTs were within 50ms for both operators. The picture, however, started to change as we look at the worst 20% RTTs with Op_1 performing worse than Op_2.

We are interested in investigating whether high delays, the top 10%, can be predicted based on historical RTT values and available high-level metadata about connection quality. We intentionally avoid using cross-layer information such as MAC layer scheduling decisions and physical layer reports. This is because leveraging these in practice would require complicated APIs that can communicate with the underlying chipset, for example, the approach that tools such as MobileInsight use [11]. Hence, our problem is a classical forecasting problem, which may suggest that available time series analysis techniques such as ARIMA can be a good fit [12]. However, these methods base their prediction chiefly on past values and patterns in the time series and do not lend themselves easily to regularization, that is, adjusting forecasting by incorporating side information about relevant factors such as signal quality. Thus, machine learning (ML) appears to be a viable alternative.

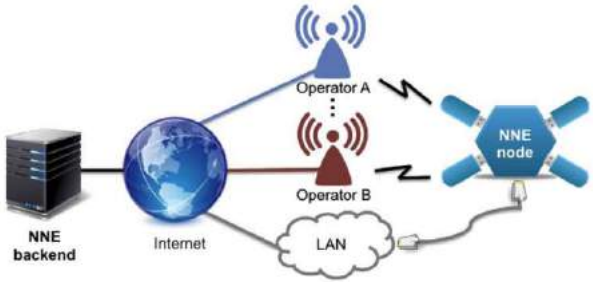


FIGURE 2. Measurement setup [10].

Figure 3 shows that attempting to predict high delays means that we need to handle a heavily imbalanced dataset. Specifically, we categorized the delays into low and high using one threshold per operator, which is 80ms and 60ms for Op_1 and Op_2, respectively. These thresholds are meant to designate the top 10% delays as high, that is, our classes have a relative ratio 9:1 by design. To prepare a balanced dataset, we investigated both oversampling and undersampling. We use the synthetic minority oversampling technique (SMOTE), which applies a nearest neighbor algorithm to generate synthetic data for the minority class [13]. For the undersampling, we used the NearMiss algorithm, which removes samples from the majority class. It removes values that are close to the minority class to increase the spacing between the two classes and avoid information loss [14]. Fitting a random forest classifier, a supervised ensemble learning method [15], and using both SMOTE and NearMiss to balance our data, yields a comparable accuracy of $\approx 79\%$. We decided to proceed with undersampling because it does not require the use of synthetic data.

B. CLASSIFICATION ALGORITHMS

As explained above, our problem is essentially a classification and prediction problem. To this end, we compare the performance of four supervised classification algorithms, which are listed as follows:

1) LOGISTIC REGRESSION (LR)

An interpretable binary classifier that uses a logistic function to model the binary variable. However, it usually does not perform well when the feature space is large [15].

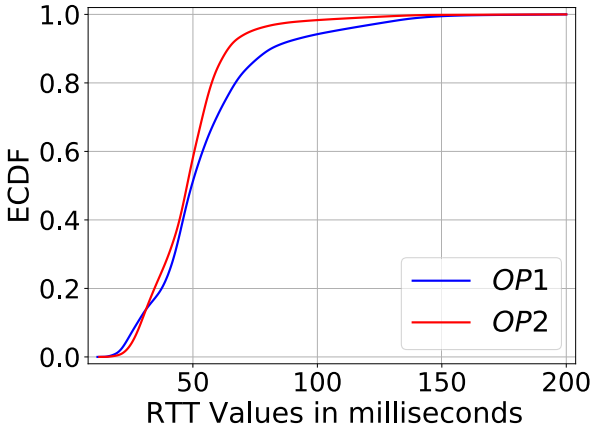


FIGURE 3. Distribution for RTT values for Op_1 and Op_2.

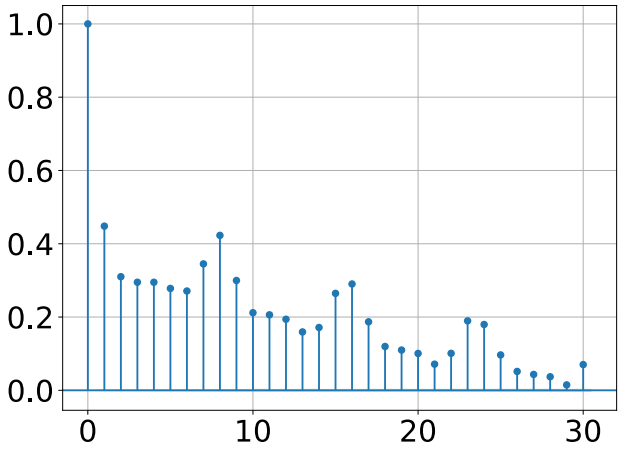


FIGURE 4. Temporal auto-correlation for RTT at time lag = 0,5,10,..,30 seconds.

2) RANDOM FOREST (RF)

An ensemble-based learning algorithm that uses many decision trees to perform either classification or regression [15]. For classification, each decision tree makes an independent prediction, which is then counted to produce the final output. RF is quite robust, but offers less interpretability than LR.

3) LightGBM

A gradient-boosting framework is based on decision tree algorithms [16]. Gradient boosting algorithms combine iteratively a number of weak learners into a single strong learner. Similar to RF, LightGBM is less interpretable than LR.

4) ENSEMBLE

This approach combines the logistic regression, LightGBM, and random forest classifiers into a single model [17]. Each algorithm is trained separately. Then, a gradient-boosted decision tree is trained, based on the predictions from each algorithm along with the input data. This allows for weighting the contribution of each classifier, resulting in a combination that is an improvement over the individual classifiers.

C. RELEVANT FEATURES

We used four groups of features to train the classifiers. The guiding principle in picking these features is to limit ourselves to features that can be readily available to applications and minimize dependencies on cross-layer features. The four groups comprise radio reception quality, diurnal, spatial, and time-series effects.

1) RADIO RECEPTION FEATURES

These involve RSSI, RSRP and RSRQ.

2) DIURNAL EFFECTS FEATURES

The RTT exhibits a certain periodicity in both daily and weekly patterns. To model these effects, we assign each RTT measurement to the respective hour of the day and day of the week.

3) SPATIAL FEATURES

To account for the location of the probe (e.g., urban vs. rural), we identified the coordinates of each probe and mapped it to a (1km$\times$1km) geographical unit that is provided by the state [18]. We then found the population that resides in each identified geographical unit. Based on the distribution of the population per geographic unit, we defined three categories for this feature: (i) low ($<$ 10,000), (ii) medium (10,000-15,000) and (iii) high ($>$ 15,000). These thresholds were determined based on the distribution of the country's population.

4) TIME SERIES FEATURES

We examined whether the RTT time series exhibited autocorrelation and long-range dependence [12]. Figure 4 shows the autocorrelation function (ACF) for the RTT time series from a sample connection at different lags in seconds. We recorded a non-negligible autocorrelation that spreads over several lags. The ACF became weaker for higher lags. While this may be expected because the dataset is dominated by low RTTs, it also indicates that high RTTs may have a serial pattern to them. Therefore, we investigated whether previous RTTs can help predict upcoming delays. To determine how long we need to look back at time, we evaluated the correlation between the current RTT and RTTs from the past 3, 5, 10, 15, and 30s. Limiting ourselves to the past five seconds yielded a reasonable accuracy.

D. APPROACH

We trained four classifiers, one per each of the above algorithms, using one week worth of data from Op_1. We applied these algorithms using the implementations provided by the Python library scikit-learn [19] and LightGBM [16]. We applied RF using a maximum of 600 estimators and adjusted the weights proportionally to the class size. We used K-fold (k = 5) cross-validation for the hyperparameters selection. Further, as for LightGBM, we used the gradient boosted

TABLE 2. Values of accuracy and MCC for different classifiers.

Classifier	Accuracy	MCC
Logistic Regression	64.2%	0.30
Random Forest	78.8%	0.58
LightGBM	76.1%	0.53
Ensemble	80.0%	0.60

decision tree algorithm with a learning rate of 0.01. Owing to the large dataset size, we chose a larger learning rate to reduce the required number of iterations.

We used the first week of September for training and the second week for validation and evaluation. Furthermore, we only focused on Op_1 when fitting the model and used the Op_2 dataset to check whether the model generalizes to other operators.

IV. PREDICTION PERFORMANCE

We now proceed to evaluate the performance of the aforementioned classification algorithms. To this end, we investigate their general accuracy, as well as their efficacy in predicting high delays and model transferability to other network operators.

A. PREDICTION ACCURACY

We applied a number of metrics to compare the four classifiers in use, which we summarize next.

- **Accuracy.** Ratio of correctly classified samples. We also present the accuracy in the form of a confusion matrix.
- **Receiver operating characteristic (ROC) curve.** A graphical measure of the separability of a binary classifier as we vary the discrimination threshold.
- **Precision-Recall curve.** The plot describes the trade-off between precision and recall for different thresholds. A high area under the curve represents both a high recall and high precision. This is more appropriate for imbalanced datasets.
- **Matthews correlation coefficient (MCC).** A measure of the correlation between the actual and predicted samples [20]. Unlike other metrics, MCC is symmetric because it assigns all classes equal importance.

In our evaluation, the true positives (TPs) are the correctly classified high-delay samples. True negatives (TNs) are the correctly classified low-delay samples. The false positives (FPs) are the incorrectly classified low-delay samples. Finally, false negatives (FNs) are the samples incorrectly classified as high-delay.

Table 2 presents the prediction accuracy and MCC of the four classification algorithms. The ensemble, random forest, and lightGBM outperformed logistic regression by a clear margin. The MCC confirms that the results of the ensemble and random forest correlate well with the actual classes across the board. The ensemble model achieved a very good accuracy compared to the closest related work by khatouni et. al [21], which achieved an accuracy of 67% when applying DT using the same features defined by the authors. We believe that the lower accuracy of [21] is due to having neglected the effect of historical data in their model.

The ROC curves and the respective area under the curve (AUC) values in Figure 6 further confirm the above observations for a range of thresholds. The ensemble classifier outperformed all the other three classifiers, with an AUC value of 0.88. The random forest classifier had an AUC of 0.87. The corresponding values for the lightGBM and logistic regression are 0.84 and 0.69, respectively. Moreover, as expected the precision-recall curves in Figure 7 show similar results.

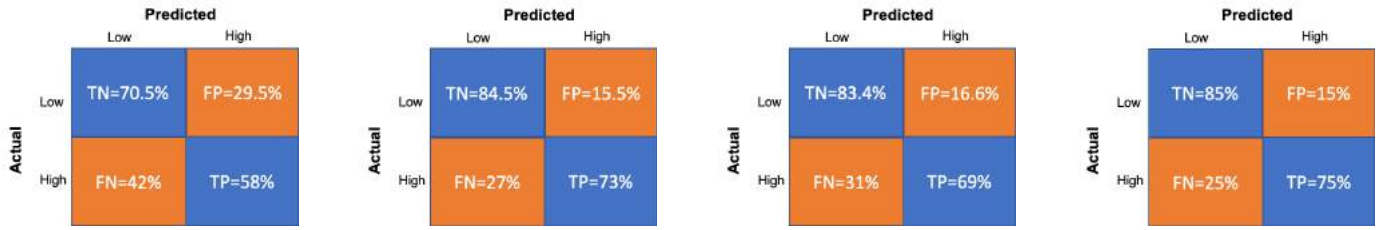
The confusion matrices for the four classifiers (see Figure 5) further confirm that the ensemble and random forest predict both the TPs and TNs with reasonable accuracy, although the performance is marginally worse when predicting high delays. The ensemble classifier correctly predicted 75% of the high-delay samples. Although this is a relatively good accuracy, we need to investigate whether the model can accurately predict high jumps in delay, as these have the worst impact on end-to-end performance.

B. FEATURES IMPORTANCE

To gain more insight into our model, we identify the features that contribute the most to the decision-making of the model. Figure 8 presents the top 10 features, along with their importance. Historical RTTs, taking the first five spots, play the most important role. Additionally, the network features RSSI, RSRP, and RSRQ contribute to discriminating features in the model. We evaluated our model when relying on historical RTTs only; the ARIMA model used a lag order of 5, resulting in 63% accuracy. Therefore, a model that uses only historical RTT data (e.g., moving average, exponential smoothing, and ARIMA [12]) does not work well. Our model also exhibits some spatial dependencies through the population feature based on the probe location.

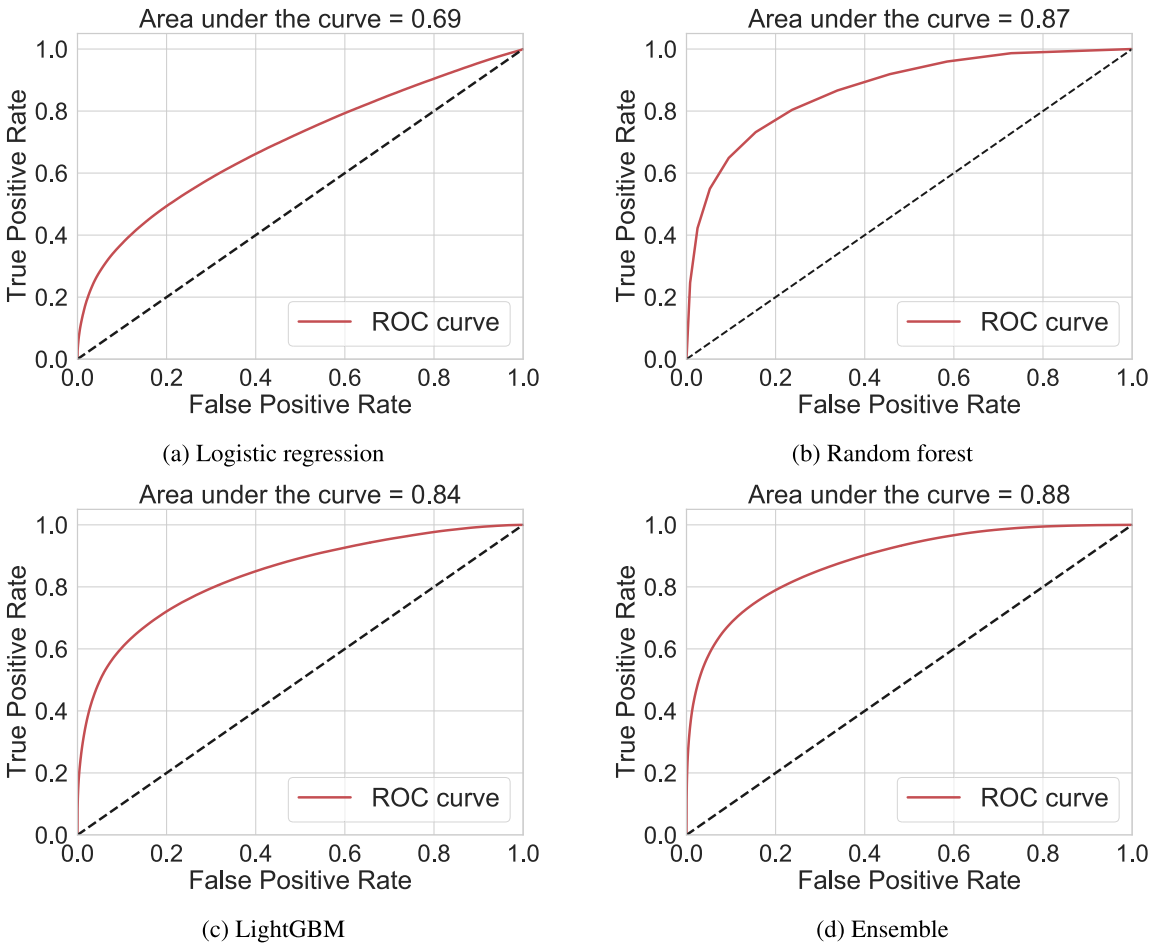
C. MODEL ACCURACY PER PROBE

We now break down our analysis of accuracy per probe, which should provide a more fine-grained idea about the failure of the model. Figure 9(a) depicts the fraction of FNs per each probe, which is the fraction of high delays that are incorrectly predicted. We observed marked differences between the probes. While a sizable majority had an FN rate below 0.1, eight probes had a rate over 0.3. However, there are fewer variations in FPs across probes (see Figure 9(b)), with accuracy below 0.2 for almost all probes. To gain insights into the high variability in the FN rate, we compared the distribution of the number of consecutive seconds with high delays for the two probes with the highest (probe 8) and lowest (probe 12) FN rates. This comparison is motivated by the fact that past RTT values are the most central features. The results show that the node with the lowest FN rate suffers longer periods with high delays as opposed to the node with the highest FN rate. Here, 90% of high-delay episodes last two seconds or shorter. Furthermore, probes with higher FN rates are generally characterized by



(a) Logistic regression (b) Random forest (c) LightGBM (d) Ensemble

FIGURE 5. Confusion matrix for four classifiers.



(a) Logistic regression (b) Random forest (c) LightGBM (d) Ensemble

FIGURE 6. Comparative evaluation of four classifiers based on ROC curve and AUC.

lower delays and less variations in delay, whereas those with medium and lower FN rates suffer higher delays. Accordingly, the classifier fails to predict on-off hikes in delay but performs well for connections with a challenging delay profile.

D. MODEL TRANSFERABILITY

Many machine learning models are limited to a specific context, which necessitates building new models as the context changes. Hence, an important question is whether our classifier is transferable to other network operators. To verify this, the model was used to predict delays for probes from the second operator Op_2 in our dataset, while training it on data from Op_1. Note that all the results above are for Op_1. A blind application results in a poor accuracy of 63%. The main reason for the performance degradation is that the two operators have different delay profiles (see Figure 3). Recall that Op_2 exhibits lower delays with 90% of RTTs lower than 60ms, while the corresponding number for Op_1 is 80ms. Accordingly, when we changed the threshold that separates low and high delay for Op_2 to 60ms, the accuracy of the model increased to 81%, which is similar to the Op_1's case. Figure 10 shows the prediction recall and ROC curves for Op_2, which closely match the corresponding plots for Op_1.

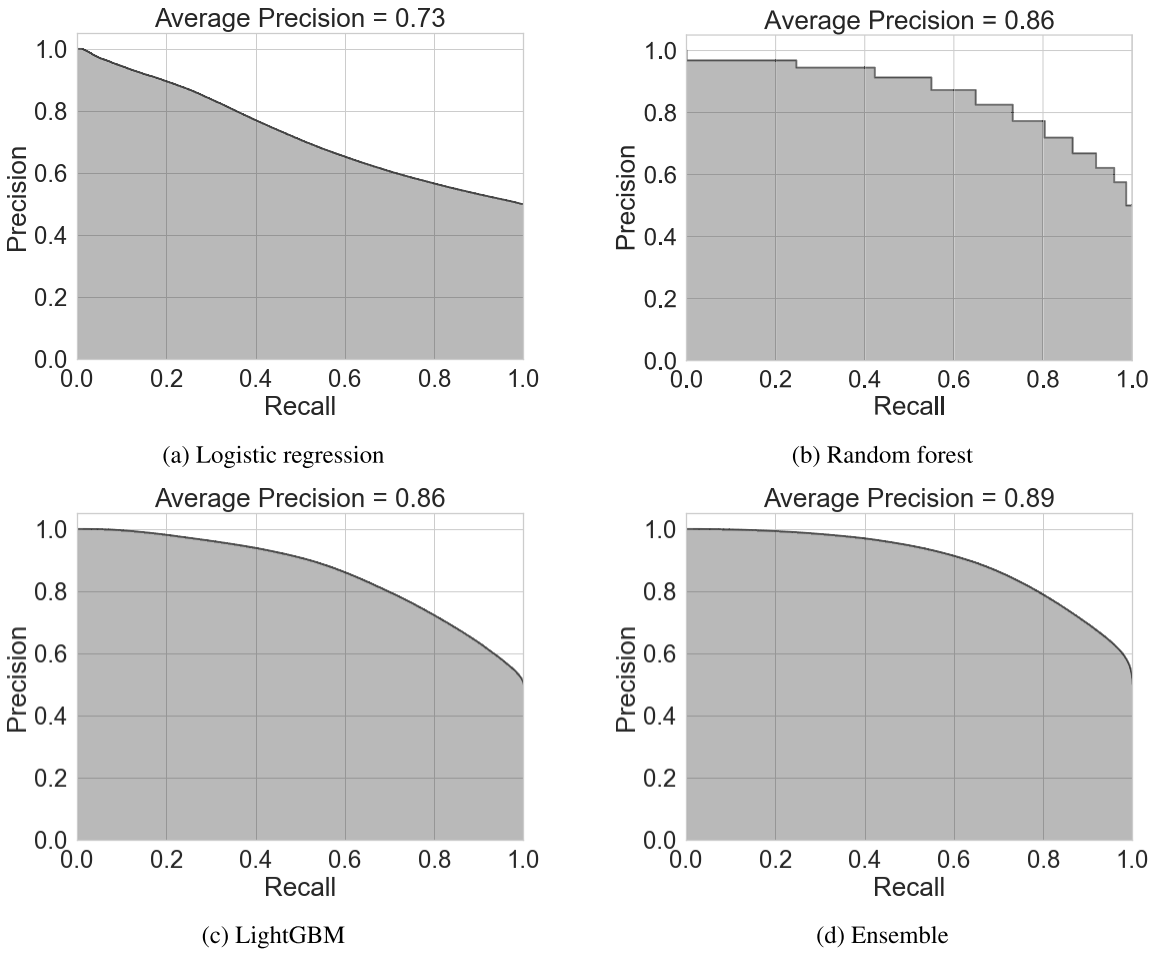


FIGURE 7. Comparative evaluation of four classifiers based on Precision-Recall curve and the average precision.

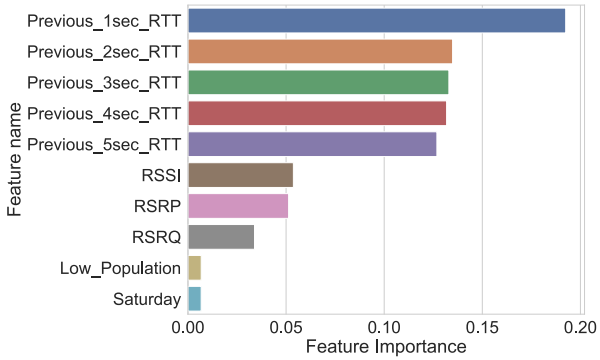


FIGURE 8. Top 10 important features using random forest.

This shows that the model is transferable once it is adjusted to the profile of the new operator.

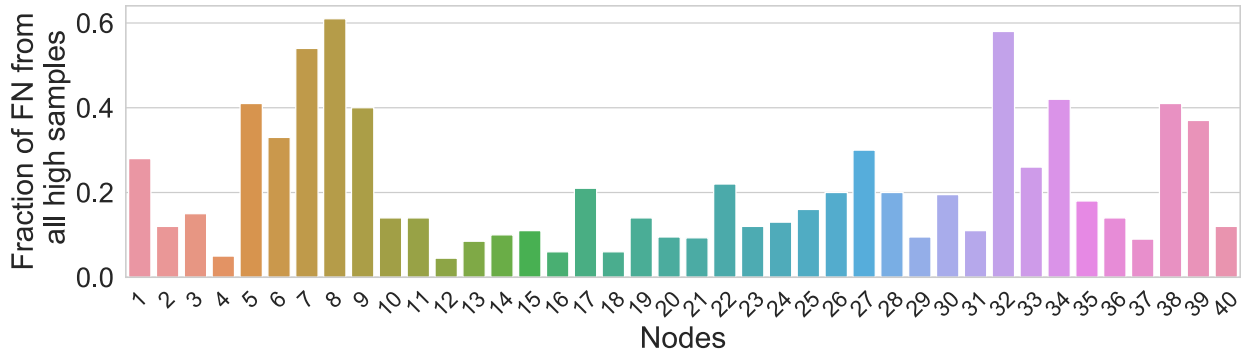
Takeaways. A simple machine learning classifier can predict fairly well whether future delays will be over or below a specific threshold. Our ensemble learning classifier is accurate in 80% of the cases and is able to predict 75% of high-delay instances. Recent RTTs, signal quality, and number of users are the most important discriminating features. Furthermore, the accuracy of the model varies across probes and is a function of the delay profile of the probe. The model is more accurate for probes with high-delay episodes that last longer. Finally, the model is transferable to other contexts that require only minor adjustments.

V. DISSECTING AND INTERPRETING THE MODEL PERFORMANCE

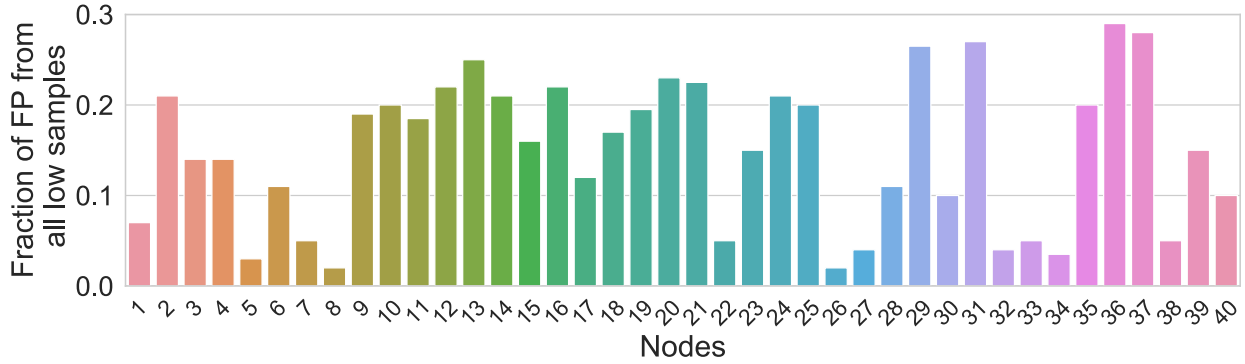
Having seen that high delays can be predicted with a reasonable accuracy, we dig deeper into the misclassified instances. Overall, our model misclassified 20% of the tested samples. These include both the FPs and FNs, which we investigate next. More specifically, we examine the high-importance features of the misclassified samples in comparison with the correctly predicted ones.

A. FALSE NEGATIVES

Recall that by FNs we refer to high delays that are incorrectly predicted as low delays. This is approximately 25% of the total high delays. Considering that historical RTT values are the most important features in our model, we compared the distribution of the previous 1-second and 2-second RTTs for the FNs with those for the TPs. The left panel in Figure 11(a) illustrates that the previous second RTT is evidently higher

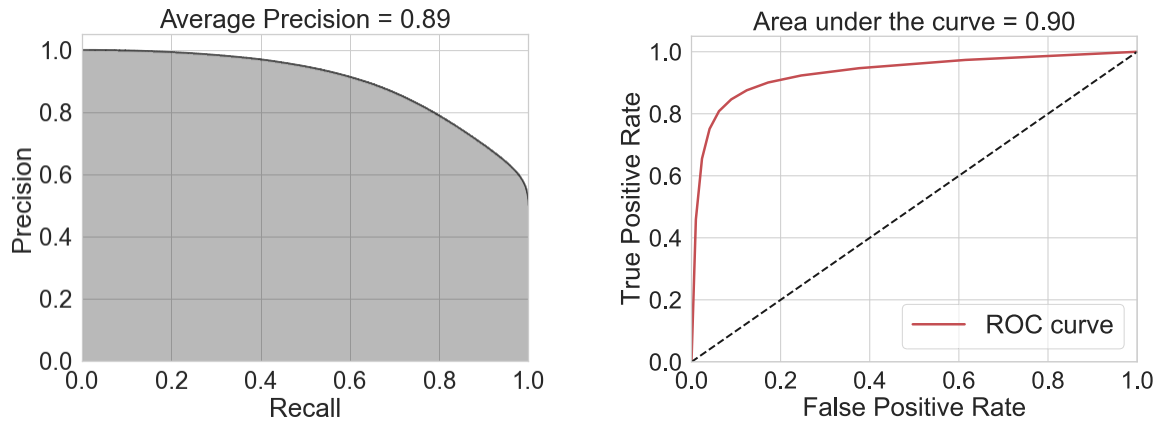


(a) The fraction of the false negatives per probe



(b) The fraction of the false positives per probe

FIGURE 9. Distribution of false negatives and false positives per probe.



(a) (b)

FIGURE 10. Prediction-recall curve and ROC curve for Op_2.

for the TP case. The previous second RTT was in the low category for 90% of FNs, as opposed to 70% of TPs. The right panel shows that, unlike TPs, FNs are often followed by high RTTs. We also compared the distribution of radio metadata (i.e., RSSI, RSRP, and RSRQ) for FNs and TPs, which are almost identical, indicating that differences in these features do not offer further details to explain FNs.[3]

[3]We do not include the respective figures due to space limitations.

Recall from the previous section that FNs appear to increase as the duration of high-delay episodes decreases. To confirm this, we plotted the accuracy as a function of the delay episode duration, as shown in Figure 12. The results indeed confirm the earlier observations, our model predicted only 57% of the high episodes of length one, i.e., spikes and 63% of length two episodes. The accuracy continues to improve as the duration of the high-delay episodes increases. Accordingly, FNs have two characteristics: 1) they often

appear after periods with low delay. That is, an FN sample may be the first sample with a high delay, and 2) they belong to delay episodes that are short.

Interestingly, our model still succeeds in predicting a considerable fraction of delay episodes of lengths one and two, which begs the question of what features help discriminate these short delays. Looking at all features, we found that the RSRQ is the most important discriminator.

Figure 13(a) compares the RSRQ distribution for the cases of length one. The TPs were evidently associated with worse RSRQ. We refer to RSRQ values as good or bad according to LTE RSRQ mapping table defined by 3GPP in [22]. We also analyzed high-delay episodes that lasted for two seconds. Here, we have four cases: 1) both delay samples are predicted correctly as high and they contribute to the true positives, 2) both delay samples are predicted incorrectly as low, 3) the first sample is predicted correctly and the other one is not and 4) lastly the second sample is predicted correctly, and the first one is not. For the first case, both samples have relatively low RSRQ values (see Figure 13(b)), which drives the model to predict them correctly as high. In the second case, both samples have a good or fair RSRQ value (see Figure 13(c)). For case three, both RSRQ values are relatively low, which explains why the first sample is predicted correctly (see Figure 13(d)). Finally, for the fourth case, the RSRQ values are higher for the first sample than for the second one (see Figure 13(e)). These results indicate that RSRQ reliably contributes to flag short delay episodes. A worse RSRQ is indicative of a congested cell. To confirm this, we break the correctly predicted high delays that belong to short episodes of lengths one and two, down to per hour of day. Figure 14 shows this breakdown, where we can clearly see that such delays tend to occur more at peak hours.

B. FALSE POSITIVES

Similarly, we investigated the FPs by examining the distributions of historical RTTs. The plots in Figure 11(b) compare the past second and two seconds RTT for FPs and TNs. We recorded a qualitative difference between FPs and TNs, where a nontrivial fraction of FPs appears to follow high-delay instances, that is, the previous second had a high delay. Looking at the RTTs in the seconds that immediately follow an FP, reveal that these seconds are often associated with low delays.

Takeaways. Short-delay episodes are difficult to predict. RSRQ helps identify short episodes that are likely to be caused by congested cells. These amounts to 57% of the episodes of length one. More frequent measurements, that is, at a frequency less than 1*s*, can help in predicting false negatives. In addition, the model struggles to demarcate the ends of some delay episodes, resulting in false positives.

VI. MODEL STABILITY AND NEED FOR RETRAINING

As machine learning-based models are trained on data collected from the past, they often degrade over time owing to external changes in the environment. Model degradation is often rectified through a system for retraining, that is, keeping the model up to date through training on the data collected during production. In this section, we investigate whether our model remains stable over time and how to rectify performance degradation, if any. Figure 15 shows the performance of an ensemble model trained on the first week of September 2018, which was then tested on data for the following seven weeks, which is the second half of September and the whole of October. The graph shows a clear trend of performance degradation, where the accuracy drops from 73% to 68%. Hence, it is necessary to retrain the model.

To gain insight into how a model may degrade over a small period of time, we monitored the performance of a model trained on the first week of September 2018 and deployed over the following seven weeks. Figure 15 shows the results for a model that was not retrained, a model trained every week, and a model trained every day. Each retraining session used data collected from the previous training session. We observe that a model that does not perform any retraining exhibits a downward trend in performance. Retraining every day shows an improvement over not retraining but still has a downward slope. We believe this is because the model does not have sufficient training data to accurately represent day-to-day changes in mobile delay. Retraining every week shows an improvement over retraining every day, and now shows a slight upward trend in performance. The experiments show that retraining a model helps stabilize the performance but may still change from day to day due to unforeseen circumstances. An example of such a change can be seen on October 15th, where the network had a surge in dropped packets and high delays due to the failure of a central component.

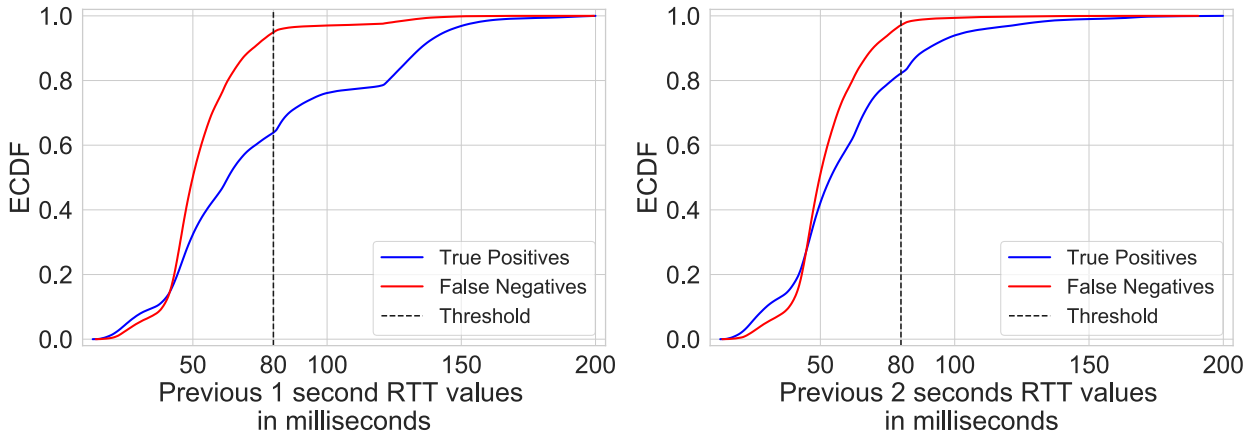
Takeaways. The model performance degrades as time progresses. As expected, retraining can help address this. However, the retraining cycle must be adjusted to include all important patterns in the underlying dataset. We found that a modest weekly cycle performed fairly well.

VII. ARE HIGH DELAYS ALSO PREDICTABLE IN 5G?

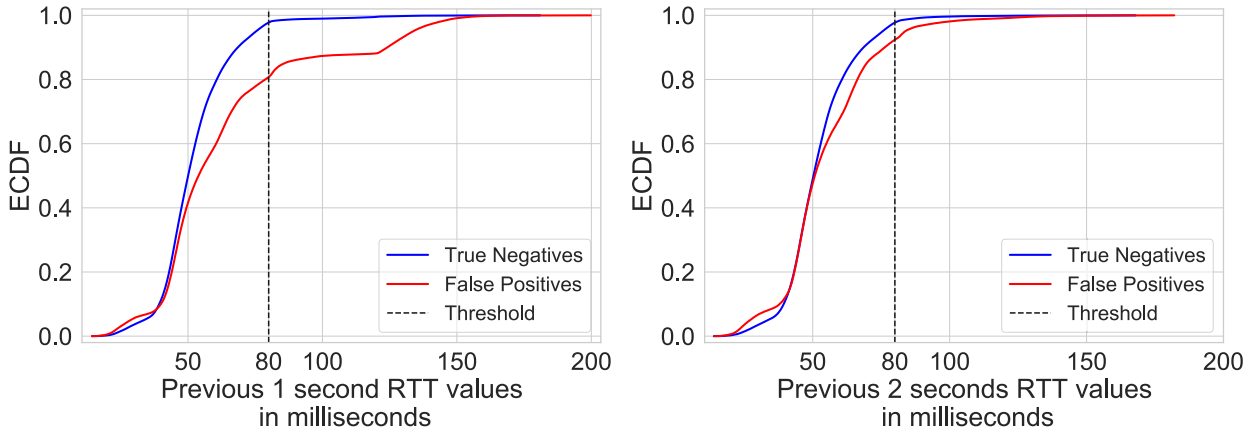
In this section, we examine whether our ensemble classifier can be extended to 5G.

A. DATASET

We collected RTT measurements, following a procedure similar to that for 4G, using three measurement nodes that connect to the newly launched sub-6GHz Non-Stand Alone (NSA) 5G [23] service by *Op*1. These nodes connect to the 5G network using Huawei CPE Pro 2 [24] and commercial subscriptions. Similar to the 4G nodes, the three nodes were placed indoors in an urban environment. In addition to active measurements, the probes collect the connection metadata. The collected metadata involves the same 4G metadata described in Section II and a set of extra metadata. The additional metadata include the modulation coding schemes (MCS) in use (i.e., the number of bits that can be sent in a resource block for both uplink and downlink [25]),



(a) False negatives and true positives



(b) False positives and true negatives

FIGURE 11. The distributions of historical delay features (previous 1 second RTT and previous 2 seconds RTT values) for the false negatives, true positives, false positives and true negatives.

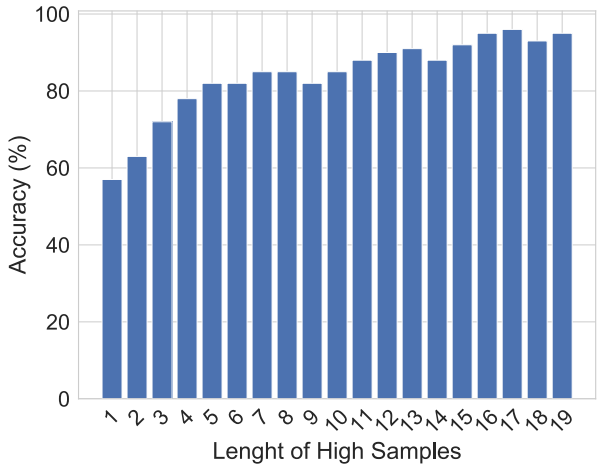


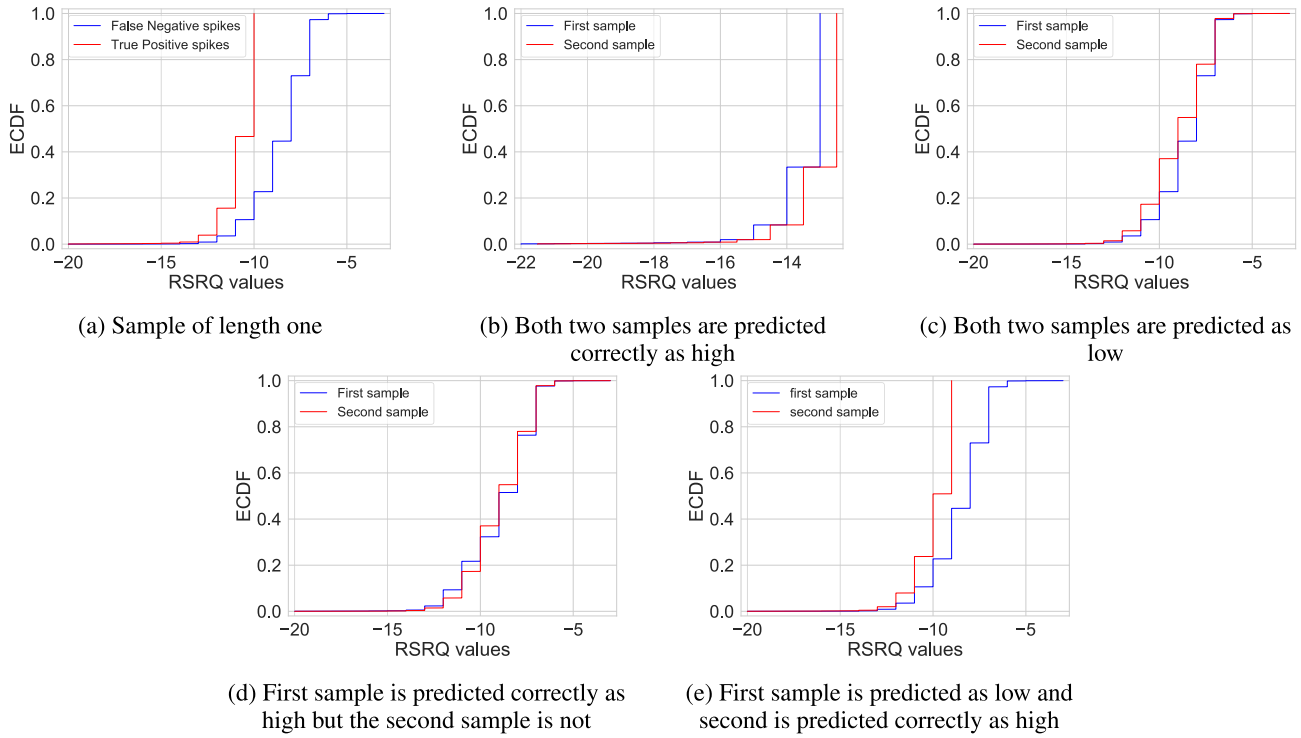
FIGURE 12. Accuracy of the high delay episodes grouped by how long the episode lasts.

the measured power on the physical uplink shared channel, the physical uplink control channel, and the power of the sounding reference signal. The MCS captures the quality of the downlink, whereas the three power measures capture the quality of the uplink. Measurements were conducted in February 2020. After merging the RTT measurements with metadata, we obtained a dataset containing 838,796 samples.

Figure 16 shows the RTT distribution for the 5G data *Op*1. The top 10% RTTs, categorized as high delays, correspond to RTTs exceeding 28*ms*. This is a major improvement over 4G. Note that the 5G NSA still uses the 4G core, albeit with a flattened architecture. However, it deploys a different air interface, that is, 5G New Radio. The measured network delivers 5G NSA over a number of frequencies, but only focuses on the commonly used 3.5 GHz frequency. The flattened architecture and differences in the air interface explain most of the savings in RTT [5].

B. PREDICTION ACCURACY

To verify whether high delays can be predicted equally well on 5G as in 4G, we trained an ensemble classifier with three weeks 5G data and tested it using a one-week dataset. Further, we conducted two experiments: one using the same features as for the LTE in Sec III and another using the extra metadata as features. The first experiment resulted in an accuracy of 83%, while the second experiment achieved an accuracy



(a) Sample of length one

(b) Both two samples are predicted correctly as high

(c) Both two samples are predicted as low

(d) First sample is predicted correctly as high but the second sample is not

(e) First sample is predicted as low and second is predicted correctly as high

FIGURE 13. The distributions of RSRQ of high delay episodes of lengths one and two.

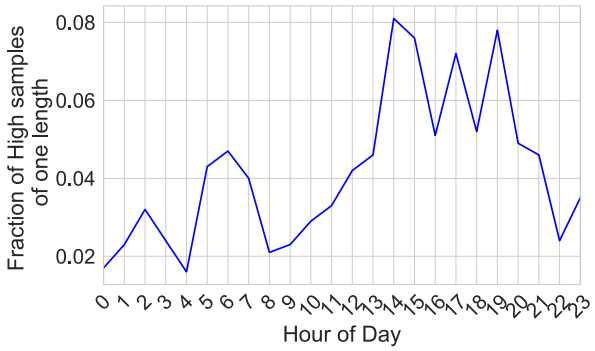


FIGURE 14. The diurnal pattern of short high delay episodes.

of 88%. Figure 17 shows the performance of the second experiment. The model had high precision and an AUC close to 1. From the confusion matrix in Figure 17(c), the model can successfully predict approximately 85% of the high-delay instances.

C. FEATURES IMPORTANCE

Figure 18 presents the top 15 features along with their importance for the 5G model. As for 4G, historical RTTs ranked as the top five important features. Next comes the downlink MCS and RSRP. Note that all the new features bring a non-negligible contribution to the model.

D. MODEL TRANSFERABILITY TO 5G

In Sec. IV, we verified that the model is transferable between operators in the case of 4G data. However, when we evaluated the 4G model on 5G data, this resulted in a significant drop in accuracy of 51%. Figure 19 shows the density plots for 4G data from *Op*1 and *Op*2, which are quite similar and explain why the model is transferable between *Op*1 and *Op*2. On the other hand, the density plot for 5G data for *Op*1 is completely different from the 4G data. Thus, the model from 4G data cannot be used for 5G data. New models should be trained on 5G data to achieve high prediction accuracy.

Takeaways. The ensemble classifier works very well on 5G data, although the distribution of RTT values is very different from that of 4G. The model accurately predicted 85% of the high-delay cases. Enhancing the set of features increases the model accuracy from 83% to 88%, which is a reasonable improvement.

VIII. RELATED WORK

Existing studies have explored several approaches for predicting delay. The authors in [26] discussed different RTT prediction systems and classified them into three classes: a) localisation measurement systems; which use direct RTT measurements to form a structured overlay network to predict RTT, b) network coordinate systems; which use the geometric space to position the actual RTT measurement in order to predict RTT without direct measurement, and finally, c) matrix factorisation systems; that solve a large distance matrix in order to predict RTT. Further, they reviewed the performance, robustness, and security of these system. In the context of Internet, [27] surveyed some techniques for end-to-end Internet delay prediction, including *the time series*

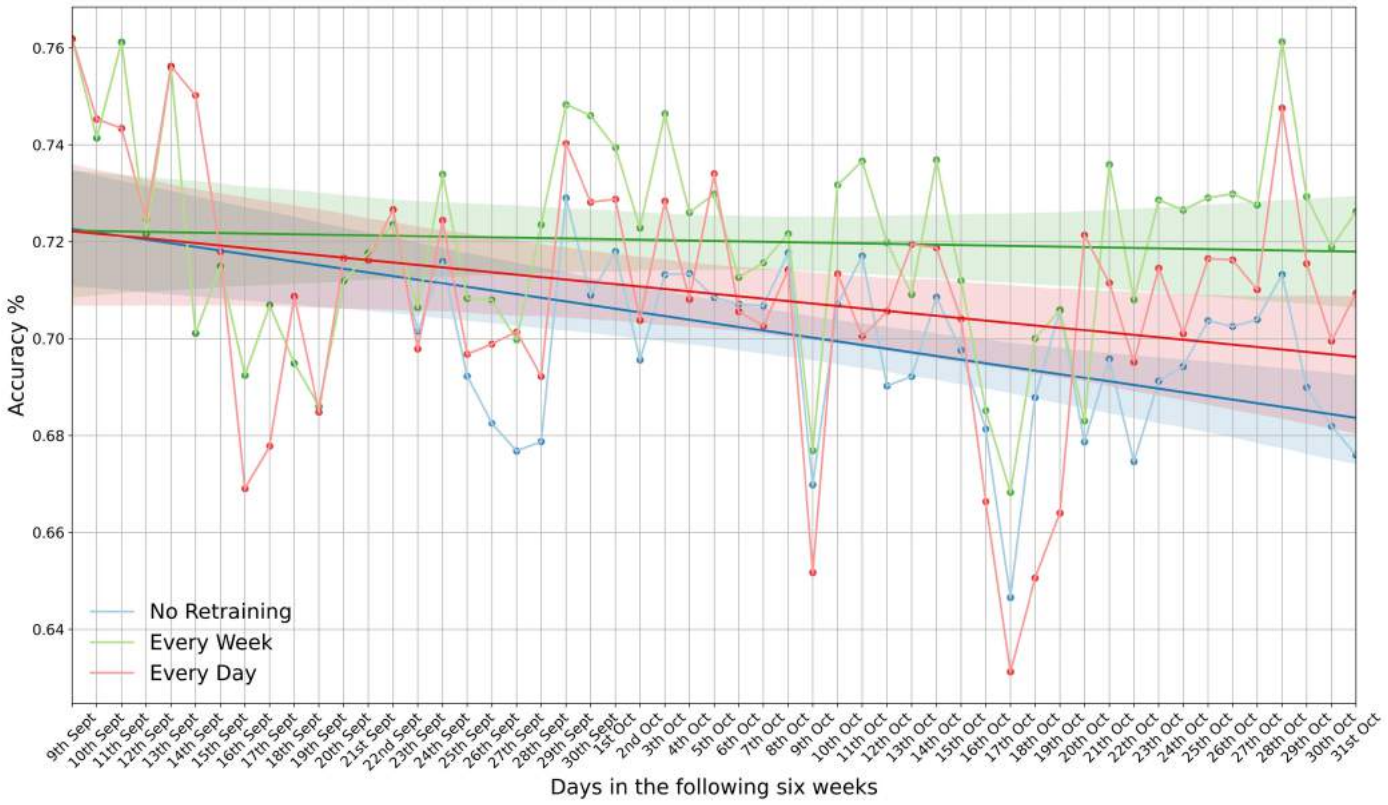


FIGURE 15. The performance of three ensemble models trained on the first week of September 2018 and evaluated over the following seven weeks. One model was not retrained over the evaluation period, one model was retrained every day, and one model was retrained every week.

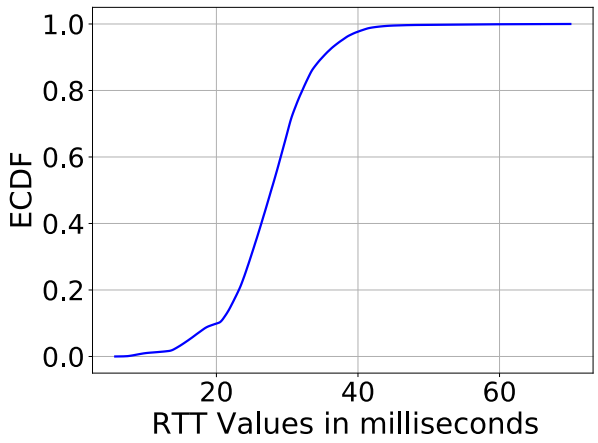
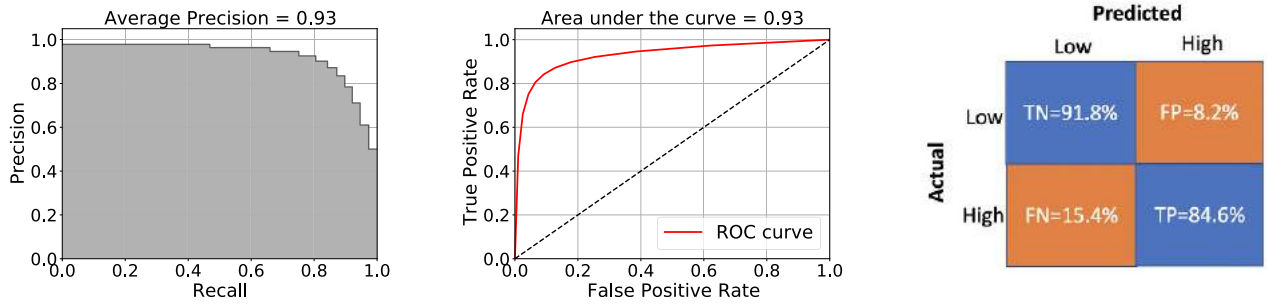


FIGURE 16. Distribution for RTT values for 5G data.

approach, queuing theory, machine learning, and neural networks. Time-series approaches using the autoregressive moving average (ARMA) models have been widely used to predict RTT. [28] presented an autoregressive eXogenous (ARX) model to study the variations in end-to-end packet delay on the Internet. Although, [28] succeeded in using the ARX model to formulate the Internet delay dynamics as a control engineering problem, [29] showed that the ARMA model as a linear time-invariant model is not suitable for predicting the Internet delay owing to the high variations in delay. Further, [30] used a machine-learning technique known as ''Experts'' framework to estimate the RTT, each of several ''experts'' provides an estimated value. The weighted average of these estimated values is used to estimate the final RTT, with the weights updated after every RTT measurement.

Recently, deep learning methods that use historical data have been increasingly used for network performance prediction. In the context of delay prediction, [31], [32] used recurrent neural networks (RNNs) to model and predict Internet delays. Although RNNs have proved to be very helpful in understanding delay dynamics, the long training time makes them unpopular for online prediction. To overcome the long training time, [33] proposed a new RNN approach with a minimal gated unit (MGU) to capture temporal features of RTT and reduce the computing cost. The proposed RNNs achieved a root mean square error (RMSE) of 1.543. In addition, [34] presented a hybrid neuro-fuzzy approach for client-cloud server communication round-trip time (RTT) prediction, achieving an accuracy of 79.36%. [35] presented a Markov model with two states to predict the probability density function of RTT instead of the actual value in LTE and WiFi networks. Also, [36] proposed a machine learning regression model to predict RTT for TCP in LTE networks. Then, they discussed how such a model can be used to enable more reliable scheduling between multiple communication paths in the field of automated vehicles. The above studies used different datasets and different ways of formulating the



(a) (b) (c)

FIGURE 17. Prediction-recall curve, ROC curve, and confusion matrix for 5G data.

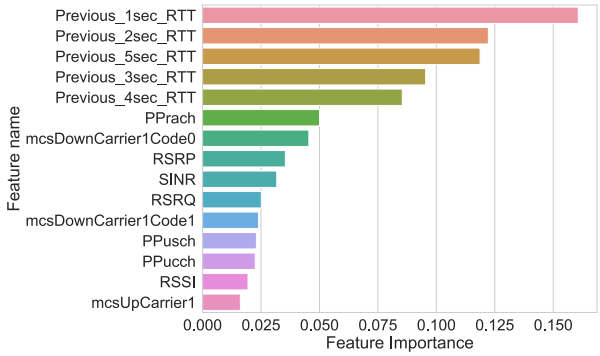


FIGURE 18. Top 15 high important features using random forest for 5G data.

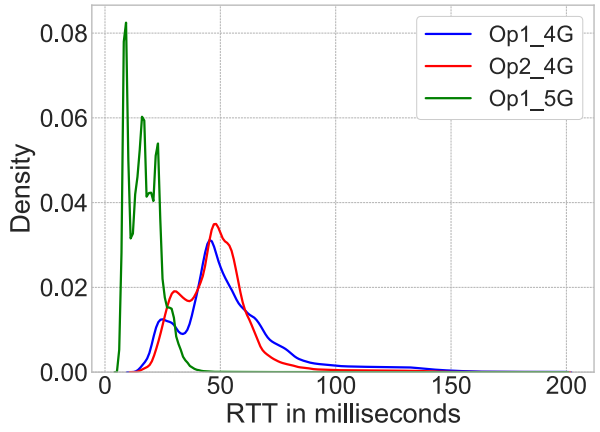


FIGURE 19. Density plots for RTT values for 4G and 5G data.

problem of RTT prediction, which do not allow for direct comparison of results. However, the approach proposed in our paper is close to the work in [21], which used traditional machine learning to predict the latency in mobile broadband networks. [21] revealed that the use of SVM, DT, and LR models to predict RTT in mobile broadband networks does not show high efficiency. The results show that performance of 71% (F1-score) when using DT. In our work, we improve the machine learning model by using an ensemble of different classifiers and incorporating historical RTTs as features. Further, we investigate the model transferability to 5G and identify the cases in which the model fails.

IX. DISCUSSION

We built an ensemble model that can predict the occurrence of high delays with 75% accuracy in 4G data. With high delays, we refer to the worst 10% delay. In the presence of multiple network interfaces that connect to different independent operators, the proposed model can be used to decide which interface to use for sending the next packet (i.e., by a protocol such as MPTCP or multipath QUIC). As a result, we can ensure an RTT within 60*ms* or 80*ms*, depending on the operator, for up to 97.5% of the time. This is a marked improvement over the default of 90%. We extend these results to 5G; by using extra metadata, we can predict 85% of the high delays. Our model captures the relationship between the misclassified samples and the duration of the high-delay episodes. High-delay instances that last for one second are contributing to almost 50% of the high-delay samples that were incorrectly predicted. To improve the accuracy, we need more fine-grained measurements (e.g., every 100*ms*) for RTT to detect such short episodes. However, this implies a trade-off between accuracy and measurement overhead, which we would like to explore in future work.

We identified two key properties of such delays as side products for predicting high delays. First, they tend to cluster time, and second, a non-trivial fraction is related to congested radio links. The clustered nature of high delays suggests that applications with strict delay requirements may need to consider multi-connectivity. A limitation of our model is that it cannot be directly applied to mobility cases. Nevertheless, we believe that many use cases with stringent delay requirements are stationary (e.g., smart meters and Industry 4.0) [1]. Delays for moving users are strongly influenced by handovers and channel fading [37]. The handover decision is highly controlled by RSRP and RSRQ levels [38]. It is not clear whether only these features can accurately help in predicting high delays under mobility. In the future, we plan to investigate this issue.

For our model to be useful, it must be deployable on end devices with relatively limited resources. This implies that we cannot opt for deep learning approaches that perform well in time-series prediction tasks such as recursive neural networks (RNNs). Our model has the ability to recognize temporal patterns without the need to manually craft complex

high-level features. For example, when using random forest in our first dataset training, the size of the single tree saved to the hard drive is approximately 0.6 MB. The memory required for neural network solutions depends on the total number of parameters, gradient, and activation. Recently, [39] presented a comprehensive assessment of the trade-offs between the performance of various machine learning models for binary classification datasets. Their results confirm that our selection of traditional machine learning methods can be more effective in terms of memory and CPU than deep learning-based neural networks.

X. CONCLUSION

We empirically investigated whether RTTs in mobile broadband networks could be accurately predicted. Using measurement data from a large number of probes, we found that a binary ensemble learning-based model can accurately predict delay classes 80% and 88% of the time for 4G and 5G, respectively.

The model is both interpretable and transferable. Furthermore, the model does not require extensive retraining but rather a modest retraining with a weekly cycle. However, it struggles when predicting short delay episodes and, to a lesser extent, by demarcating the end of a delay episode. Despite this, the model performs fairly well. For example, an application using it to anticipate high delays (i.e., the worst 10%) should be able to react positively to 75% of them when using a 4G connection.

Our findings are encouraging and can help inform the scheduling of multipath transport protocols that aim to bound delays. Next, we plan to implement our model in a multipath scheduler and investigate the means to improve the detection of short-lasting delay episodes. In addition, we plan to explore modelling scenarios with high mobility.

REFERENCES

[1] M. Iwamura, "NGMN view on 5G architecture," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, May 2015, pp. 1–5.
[2] Y. Li, Z. Yuan, and C. Peng, "A control-plane perspective on reducing data access latency in LTE networks," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2017, pp. 56–69.
[3] N. Larson, D. Baltrunas, A. Kvalbein, A. Dhamdhere, K. Claffy, and A. Elmokashfi, "Investigating excessive delays in mobile broadband networks," in *Proc. 5th Workshop All Things Cellular, Oper., Appl. Challenges*, Aug. 2015, pp. 51–56.
[4] S.-Y. Lien, S.-L. Shieh, Y. Huang, B. Su, Y.-L. Hsu, and H.-Y. Wei, "5G new radio: Waveform, frame structure, multiple access, and initial access," *IEEE Commun. Mag.*, vol. 55, no. 6, pp. 64–71, Jun. 2017.
[5] D. Xu, A. Zhou, X. Zhang, G. Wang, X. Liu, C. An, Y. Shi, L. Liu, and H. Ma, "Understanding operational 5G: A first measurement study on its coverage, performance and energy consumption," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.*, Jul. 2020, pp. 479–494.
[6] A. Elmokashfi, D. Zhou, and D. Baltrünas, "Adding the next nine: An investigation of mobile broadband networks availability," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2017, pp. 88–100.
[7] B. Han, F. Qian, S. Hao, and L. Ji, "An anatomy of mobile web performance over multipath TCP," in *Proc. 11th ACM Conf. Emerg. Netw. Exp. Technol.*, Dec. 2015, pp. 1–7.
[8] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure, *TCP Extensions for Multipath Operation With Multiple Addresses*, document RFC 6824, 2013.
[9] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proc. 13th Int. Conf. Emerg. Netw. Exp. Technol.*, Nov. 2017, pp. 160–166.
[10] A. Kvalbein, D. Baltrūnas, K. Evensen, J. Xiang, A. Elmokashfi, and S. Ferlin-Oliveira, "The nornet edge platform for mobile broadband measurements," *Comput. Netw.*, vol. 61, pp. 88–101, Mar. 2014.
[11] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *Proc. 22nd Annu. Int. Conf. Mobile Comput. Netw.*, Oct. 2016, pp. 202–215.
[12] G. E. Box, G. M. Jenkins, G. C. Reinsel, and G. M. Ljung, *Time Series Analysis: Forecasting and Control*. Hoboken, NJ, USA: Wiley, 2015.
[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, no. 1, pp. 321–357, Jan. 2002.
[14] S.-J. Yen and Y.-S. Lee, "Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset," in *Intelligent Control and Automation*. Berlin, Germany: Springer, 2006, pp. 731–740.
[15] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*, vol. 112. New York, NY, USA: Springer, 2013.
[16] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "LightGBM: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Red Hook, NY, USA: Curran Associates, 2017, pp. 3146–3154. [Online]. Available: http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
[17] R. Polikar, "Ensemble learning," in *Ensemble Machine Learning*. Boston, MA, USA: Springer, 2012, pp. 1–34.
[18] *Maps and Geodata From Statistics Norway*. Accessed: Nov. 1, 2019. [Online]. Available: https://www.ssb.no/natur-og-miljo/geodata
[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.
[20] D. Chicco and G. Jurman, "The advantages of the Matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation," *BMC Genomics*, vol. 21, no. 1, p. 6, 2020.
[21] A. S. Khatouni, F. Soro, and D. Giordano, "A machine learning application for latency prediction in operational 4G networks," in *Proc. IFIP/IEEE Symp. Integr. Netw. Service Manage. (IM)*, Apr. 2019, pp. 71–74.
[22] *3GPP Specification*, document TS 36.133 Release 8, 2013. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2420
[23] *3GPP Specification*, document TR 21.915 Release 15, 2019. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3389
[24] (Feb. 2020). *HUAWEI 5G CPE Pro 2*. [Online]. Available: https://consumer.huawei.com/en/routers/5g-cpe-pro-2/
[25] Y. Wang, W. Liu, and L. Fang, "Adaptive modulation and coding technology in 5G system," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, Jun. 2020, pp. 159–164.
[26] D. Mirkovic, G. Armitage, and P. Branch, "A survey of round trip time prediction systems," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1758–1776, 3rd Quart., 2018.
[27] M. Yang, X. R. Li, and H. Chen, "Predicting internet end-to-end delay: An overview," in *Proc. 36th Southeastern Symp. Syst. Theory*, 2004, pp. 210–214.
[28] E. Kamrani, H. R. Momeni, and A. R. Sharafat, "Modeling internet delay dynamics for teleoperation," in *Proc. IEEE Conf. Control Appl. (CCA)*, 2005, pp. 1528–1533.
[29] P. X. Liu, M. Meng, X. Ye, and J. Gu, "End-to-end delay boundary prediction using maximum entropy principle (MEP) for internet-based teleoperation," in *Proc. IEEE Int. Conf. Robot. Automat.*, vol. 3, May 2002, pp. 2701–2706.
[30] B. A. A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, and K. Obraczka, "A machine learning approach to end-to-end RTT estimation and its application to TCP," in *Proc. 20th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2011, pp. 1–6.
[31] S. Belhaj and M. Tagina, "Modeling and prediction of the internet end-to-end delay using recurrent neural networks," *Proc. J. Netw.*, vol. 4, no. 3, pp. 528–535, 2009.

[32] A. G. Parlos, "Identification of the internet end-to-end delay dynamics using multi-step neuro-predictors," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, 2002, pp. 2460–2465.
[33] A. Dong, Z. Du, and Z. Yan, "Round trip time prediction using recurrent neural networks with minimal gated unit," *IEEE Commun. Lett.*, vol. 23, no. 4, pp. 584–587, Apr. 2019.
[34] R. Damaševičius and T. E. Sidekerskien, "Short time prediction of cloud server round-trip time using a hybrid neuro-fuzzy network," *J. Artif. Intell. Syst.*, vol. 2, no. 1, pp. 133–148, 2020.
[35] S. Yasuda and H. Yoshida, "Prediction of round trip delay for wireless networks by a two-state model," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2018, pp. 1–6.
[36] J. Schmid, P. Purucker, M. Schneider, R. V. Zwet, M. Larsen, and A. Höß, "Integration of a RTT prediction into a multi-path communication gateway," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.* Cham, Switzerland: Springer, 2021, pp. 201–212.
[37] H. Deng, C. Peng, A. Fida, J. Meng, and Y. C. Hu, "Mobility support in cellular networks: A measurement study on its configurations and implications," in *Proc. Internet Meas. Conf.*, Oct. 2018, pp. 147–160.
[38] M. Tayyab, X. Gelabert, and R. Jäntti, "A survey on handover management: From LTE to NR," *IEEE Access*, vol. 7, pp. 118907–118930, 2019.
[39] D. Preuveneers, I. Tsingenopoulos, and W. Joosen, "Resource usage and performance trade-offs for machine learning models in smart environments," *Sensors*, vol. 20, no. 4, p. 1176, Feb. 2020.

AZZA H. AHMED (Member, IEEE) received the master's degree from the University of Nottingham, in 2012. She is currently pursuing the Ph.D. degree with the Simula Metropolitan Center for Digital Engineering, Oslo, Norway. Her research interests include communication networks management and control, network performance optimization, network automation, and machine learning to solve networks problems.

STEVEN HICKS (Member, IEEE) received the master's degree from the University of Oslo, Oslo, Norway, in 2018, where he studied explainable machine learning for medical use-cases. He is currently pursuing the Ph.D. degree with SimulaMet, Oslo. His research interests include machine learning, explainable artificial intelligence, computer vision, and medical multimedia.

MICHAEL ALEXANDER RIEGLER received the Ph.D. degree from the Department of Informatics, University of Oslo, Oslo, Norway, in 2015. He is currently working as a Chief Research Scientist at SimulaMet, Oslo. His research interests include machine learning, video analysis and understanding, image processing, image retrieval, crowdsourcing, social computing, and user intentions.

AHMED ELMOKASHFI (Member, IEEE) received the Ph.D. degree from the University of Oslo, Oslo, Norway, in 2011. He is currently a Research Professor at the Simula Metropolitan Center for Digital Engineering, Oslo. He is also working as the Head of the Center for Resilient Networks and Applications (CRNA), which is part of the Simula Metropolitan Centre, which is funded by the Norwegian Ministry of Transport and Communication. His research interests include network measurements and performance. In particular, he focused on studying resilience, scalability, and evolution of the internet infrastructure; the measurement and quantification of robustness in mobile broadband networks; and the understanding of dynamical complex systems. Over the past few years, he has been leading and contributing to the development, operation, and management of the NorNet testbed infrastructure, which is a countrywide measurement setup for monitoring the performance of mobile broadband networks in Norway.

Article II

Ahmed, A. H., Hicks, S., Riegler, M.A., and Elmokashfi, A. (2022, August 14 - 18,). **RCAD:Real-time Collaborative Anomaly Detection System for Mobile Broadband Networks.** KDD '22: The 28thACM SIGKDD Conference on Knowledge Discovery and Data Mining.
DOI: **https://doi.org/10.1145/3534678.3539097**

RCAD: Real-time Collaborative Anomaly Detection System for Mobile Broadband Networks

Azza H. Ahmed
azza@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway
Oslo Metropolitan University
Oslo, Norway

Michael A. Riegler
michael@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway
University of Tromsø
Tromsø, Norway

Steven A. Hicks
steven@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway
Oslo Metropolitan University
Oslo, Norway

Ahmed Elmokashfi
ahmed@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

ABSTRACT

The rapid increase in mobile data traffic and the number of connected devices and applications in networks is putting a significant pressure on the current network management approaches that heavily rely on human operators. Consequently, an automated network management system that can efficiently predict and detect anomalies is needed. In this paper, we propose, RCAD, a novel distributed architecture for detecting anomalies in network data forwarding latency in an unsupervised fashion. RCAD employs the hierarchical temporal memory (HTM) algorithm for the online detection of anomalies. It also involves a collaborative distributed learning module that facilitates knowledge sharing across the system. We implement and evaluate RCAD on real world measurements from a commercial mobile network. RCAD achieves over 0.7 F-1 score significantly outperforming current state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly Detection**; • **Networks** → **Mobile Broadband Networks**.

KEYWORDS

Anomaly Detection, Hierarchical Temporal Memory, Multivariate Time series, Mobile Broadband, Collaborative Distributed Learning

ACM Reference Format:
Azza H. Ahmed, Michael A. Riegler, Steven A. Hicks, and Ahmed Elmokashfi. 2022. RCAD: Real-time Collaborative Anomaly Detection System for Mobile Broadband Networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* KDD'22, Washington, DC, USA, 10 pages. https://doi.org/10.1145/3534678.3539097

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539097

1 INTRODUCTION

Mobile network operators face an increasing demand for mobile data traffic that is paired with the need to support emerging and novel services [34, 37]. This near continuous flux of data has compounded the complexity of network operation and management [14]. Operators need to enhance current network management practices, because newer services like industrial control, connected mobility and emergency communication expect stringent service level guarantees [17, 24]. At the same time, they need to keep the operational expenditures at a moderate level. A potential approach for striking this balance is to build systems that can automatically and efficiently predict and detect performance anomalies. In this paper, we focus on the development of such a system.

Nowadays, troubleshooting in mobile networks is often conducted at a coarse granularity and mostly manually. More specifically, operators collect key performance indicators (KPIs), usually every hour, at various network elements like basestations and core servers [22]. Then, they monitor those KPIs for deviations from the norm using simple thresholds. This process is evidently slow, coarse and reactive [33]. It can flag long varying problems, like a continuously congested hotspot, but offers very little as far as the timely identification of problems is concerned. For example, Figure 1 shows two-hour long time series of per-second round trip delays (RTT), a measure of network latency, and three coverage quality indicators (RSRP, RSRQ, RSSI) that we measured at the same location in a 4G network. We mark all anomalous measurements with red circles. This figure shows that anomalies are a common occurrence and must be tracked at a much finer granularity and can simultaneously impact a number of KPIs, i.e. correlated behaviour.

Detecting anomalous behavior on a set of correlated time series signals has been an active research area in the machine learning community for a long time [7, 31]. Recent works on multivariate anomaly detection based on deep learning [4, 15, 38] show promising results on several time series tasks, however, their performance on mobile network data needs to be further studied. This is related to the nature of the mobile data which poses some difficulties for developing an anomaly detection system. First, in mobile networks a large amount of data arrives at a rapid rate. The sheer volume of

the data makes it difficult to store it in its entirety, and it is even more difficult to operate on it in real time. Second, the mobile network is a distributed infrastructure composed of variety of probes and sensors across the entire network. Most of the current anomaly detection methods aggregate all the data to a single point and process them together. Such an approach, however, may become difficult to realise or infeasible in the future due to the exponentially growing size and scale of the network. Hence, carrying out an efficient and distributed anomaly detection remains challenging. Finally, network behavior changes rapidly over time and would require frequent retraining of a static model which is specifically challenging with deep learning-based approaches.

In this work, we propose RCAD, a novel real-time collaborative anomaly detection framework for mobile networks. The system involves two components. The first is an online distributed unsupervised anomaly detection and prediction system. The second is a collaborative framework for knowledge sharing and exchanging to improve the overall system accuracy and compensate for the lack of centralization. Specifically, we leverage a recent machine intelligence algorithm called *hierarchical temporal memory* (HTM) [20] for anomaly detection. The unsupervised nature of HTM makes it able to automatically discover relationships between variables. Moreover, compared to batch-based deep learning models, the online nature of the HTM learning algorithm makes it able to continuously learn relations between the variables without needing to train it again. Inspired by the natural intelligence in humans which allows them to transfer knowledge from known problems and their solution towards unknown ones, we define a collaborative framework between the distributed probes in the network. This way, a probe that has been exposed to fewer anomalies can benefit from other probes' experiences to improve its detection accuracy. We present two different collaboration approaches to help probes deciding on whether to replace the current model: 1) threshold-based model replacement and 2) deep reinforcement learning (DRL) based model replacement. Using real world traces, we compare RCAD to several anomaly detection methods as well as investigate various approaches to realize RCAD. Our results demonstrate the superiority of RCAD and show that RCAD with DRL-based model replacement strikes a reasonable balance between accuracy and overhead.

The rest of the paper is organized as follows. Section 2 reviews existing unsupervised methods for detecting anomalies in multivariate time series. Section 3 presents our problem statement as well as an overview of hierarchical temporal memory and related collaborative machine learning methods. Section 4 sketches the design of RCAD. Section 5 evaluates the performance of RCAD. Finally, Section 6 concludes the paper.

2 RELATED WORK

Anomaly detection in time series has been widely studied in the literature. Here, we briefly review anomaly detection methods in time series by categorizing them into supervised and unsupervised.

Supervised approaches require training a binary classifier using labels of both normal and anomalous data. These methods are different from the traditional supervised classifiers in that the algorithm should able to handle the temporal information in the data. NN-DTW [5] is a popular classifier for time series that is based on the nearest neighbor algorithm coupled with the dynamic time warping similarity measure. As this classifier achieves high accuracy in time series, different supervised classifiers have been presented as ensemble models based on nearest neighbor algorithm such as Proximity Forest [27]. With deep learning advances, InceptionTime [13] has been presented as a supervised time series classification method that consists of an ensemble of deep Convolutional Neural Network (CNN) models. Despite the high performance of supervised methods in anomaly detection, these methods come with some drawbacks, where the need of annotated data and that they cannot react to changes in continuous data streams without being retrained which is costly.

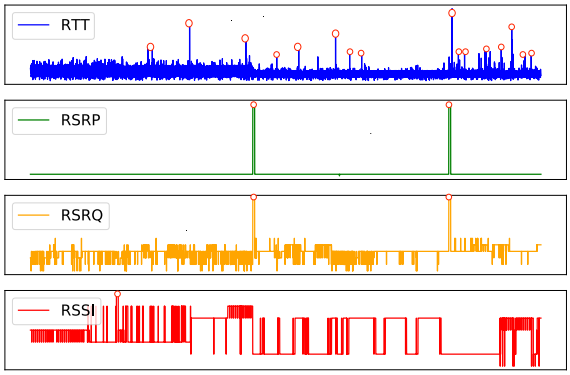


Figure 1: An example of multivariate time series data from mobile broadband network. The time series spans two hours and have one second granularity. The red circles indicate anomalies.

Different unsupervised methods have been proposed for anomaly detection such as isolation methods that focus on separating outliers from the rest of the data. Isolation Forest (IF) [26] is a popular isolation method based on decision trees. Unsupervised methods based on deep learning techniques have recently received much attention, especially autoencoder (AE) [36] based methods. These methods use the reconstruction error as the anomaly score, i.e. samples with a high score are considered to be anomalous. DAGMM [40] leverages a deep autoencoder and a Gaussian Mixture Model (GMM) to model the density distribution of multidimensional data without considering the temporal dependencies. LSTM-VAE [32] combines LSTM with a variational autoencoder (VAE); the LSTM-based encoder projects the input data and its temporal dependencies into a latent space. During decoding, it uses the latent space representation to estimate the output distribution. LSTM-VAE detects an anomaly when the log-likelihood of the current data is below a threshold. OmniaAnomaly[38] uses a variant of recurrent neural network (RNN) called gated recurrent unit (GRU) to capture temporal dependencies between multivariate observations. Then, it applies VAE for representation learning to map observations to stochastic variables. It uses the reconstruction probabilities of input samples as anomaly scores. Generative Adversarial Networks (GANs) [16] have also been used in time series anomaly detection. The GAN is trained in an adversarial way between the generator and the discriminator.

GAN-based anomaly detection uses the output of the discriminator as the anomaly score during inference. MAD-GAN [25] uses LSTMs as the base models in the GAN framework (generator and discriminator) to capture the temporal correlation of time series distributions. USAD [4] combines the advantages of autoencoders and GANs and presents an autoencoder model that is adversarially trained to amplify the reconstruction errors for anomalies so that the USAD model can detect the anomalies with small deviations from normal data. Different from previous approaches, NSIBF [15] is an unsupervised method for detecting operational failures in cyber-physical systems (CPS). This framework is designed to tackle the instability of CPS data, in which a neural network is used to capture the dynamics of CPS via a state-space model. Then a Bayesian filtering method is applied on the "identified" state-space model and anomalies are detected by estimating the probability of observed measurements over time. These unsupervised methods come with several challenges that make them less applicable in many real-world scenarios such as they need normal data for training, process data in batches and require considerable training data [6].

Recently, there has been a surge of interest in developing anomaly detection methods that are suitable for real-time streaming applications. The HTM framework is presented as time series anomaly detection for real-time applications [2]. The anomaly score depends on the prediction history of the anomaly namely forecast-based model. RADM [11] is a real-time anomaly detection based on HTM and bayesian network. The HTM detects anomalies in the univariate time series composing the multivariate time series and produces the anomaly likelihood. A naive Bayesian network takes as an input the discretized values of the anomaly likelihoods of each univariate series to assign the corresponding weights to each time series according to the structure of the multivariate series. Similarly, HTM has been used in heath care application [8, 12]. For instance, El-Ganainy et al. [12] proposed an anomaly detection framework for intensive care units in hospitals to predict the health conditions of critically sick patients based on the vital signs. It applies online learning using HTM to enable real-time stream processing and provide unsupervised predictions. These predictions are passed to an LSTM binary classifier that forecasts the status of the patient either critical or not. In this work, there are 10 time series where each one is processed separately using an HTM model to preform predictions before feeding them to the LSTM classifier as features.

In this paper, we present a novel distributed anomaly detection architecture, RCAD, that combines HTM with intelligent model exchanging between different parts of the system, i.e. agents, to maximize anomaly detection accuracy. Compared with the above mentioned approaches, RCAD can process data streams in real-time, learn the patterns of data continuously without training and improve the overall system performance by sharing the knowledge between models participating in the system. To the best of our knowledge, RCAD is the first system that proposes model replacement as a viable strategy for boosting accuracy.

3 PRELIMINARIES

In this section, we present the problem of anomaly detection in mobile networks in details and provide preliminaries about HTM and collaborative learning.

3.1 Problem Statement

In this work, we investigate the problem of anomaly detection in mobile broadband networks. To this end, we use a dataset that comprises a set of KPIs. These KPIs were collected by a set of geographically spread probes that connect to a commercial mobile network in Norway (see subsection 4.1 for details). KPIs are often correlated since they describe the behavior of the same network. By exploiting the correlation between these KPIs, we aim at producing a model that timely predicts the temporal variation of a given KPI based on the observation of other KPIs, which are considered as predictors. The end goal is to predict anomalies that concern the forecasted KPI, i.e. it exceeds a given threshold. This threshold is defined by experts or based on operator's policies.

A mobile broadband network is essentially distributed comprising a large number of basestations that are geographically spread. These basestations are controlled by a centralized core network that keeps track of subscriptions, manages mobility and connects users to the rest of the Internet. [1] This distributed nature has implications for systems that aim to predict and detect faults. Owing to the volume and velocity of the data, local decision making is the most feasible solution. However, by adopting a simple coordination between the neighbor nodes (i.e. probes that are served by the same basetstation or basetations that are close to each other) we can increase the efficiency of the system while maintaining local decision. To this end, we can employ paradigms like collaborative learning and model replacement.

Based on the above description of the problem, we can summarize the requirements for the proposed solution as the following.

- The model must make online predictions using stream data without back propagation.
- The model must learn continuously without storing the entire data stream.
- The model must run unsupervised, without manual intervention. No parameter tuning must be manually made at run-time.
- The model should predict anomalies as early as possible and minimize false positive and false negative predictions.

3.2 Basics of Hierarchical Temporal Memory (HTM)

HTM is a brain-inspired neural network model that mimics the structural properties of neocortex and the way the human brain processes information [19]. HTM is essentially a memory-based system which relies on real-time sequence learning on time-varying input streams [9]. The core of HTM is based on the principles of biological neurons and synapses. Similar to the cortex structure, the HTM network consists of multiple cortical columns, that contain cortical layers. Each layer consists of mini-columns, which represent neurons (cells) with the same perceptual field. The basic unit in HTM is the cell or neuron. A number of cells form the mini-column, and a large number of mini-columns form the network space of HTM as shown in Figure 2. A single HTM cell has two types of connections: 1) proximal connection (aggregation of feed-forward connections from the input) and 2) distal connection (aggregation

[1] The core can comprise a handful number of distinct sites.

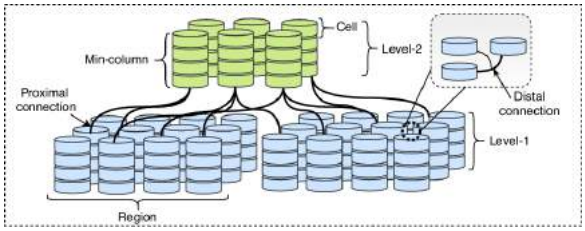


Figure 2: The hierarchical structure of HTM. The example shows 2-level HTM, where each level consists of the regions with mini-columns, and mini-columns are created from the cells.

of lateral connections from cells of the other columns). Each cell can be in three states: 1) inactive (the default state), 2) predictive, and 3) active. The predictive state of a cell is determined by the activity of the distal connections; the sum of activations weights of at least one of the distal connections exceeds a certain threshold. A cell becomes active at any time only if it was in the predictive state at the previous time instant.

Figure 3 shows the end-to-end framework for the HTM-based prediction system. Next, we elaborate on each component.

- **Encoder**. SDRs are the language of brain, therefore, HTM uses SDR to represent the input data. An SDR consists of a large array of bits where at any point in time a small percentage of the bits are ones (in blue) and the rest are zeros (in white). The encoder is responsible for determining, for a given input value, which output bits should be ones (active), and which should be zeros (inactive), in such a way as to capture the important semantic characteristics of the data. Different encoding schemes have been proposed for the different data types such as numeric encoders, geospatial encoders and natural language encoders [35].
- **Spatial pooler**. After data encoding as SDRs, the spatial pooler identifies the spatial relations between different regions in the incoming SDR and groups them together into a common output representation based on spatial similarity. The spatial pooler incorporates several computational principles of the cortex. It relies on Hebbian learning in modifying the nature of the proximal connections between mini-columns and their inputs [21]. Each connection is associated with a permanence value. The permanence values of synapses aligned with active input bits are increased, and those aligned with inactive input bits are decreased [10]. Learning happens only in those mini-columns of the spatial pooler which are active (in red), while inactive mini-columns (white) will not learn anything. The output of spatial pooler is an SDR representing the mini-columns of HTM that will be activated by the temporal memory.
- **Temporal memory**. It represents the basic component of HTM, i.e. the memory of sequences. Temporal pooling enables us to understand the sequential pattern over time. It learns the sequences of the active mini-columns from the spatial pooler and determines the activated and predicted cells. The temporal memory activates a cell if its mini-column is active and it is in predictive state and deactivates the others. If none of the cells in an active mini-column is in a predictive state, all cells in this mini-column will be activated. When a predictive cell receives a new input, the synapses on its connections are reinforced through Hebbian learning. For any such connection with few synapses, new synapses grow to a randomly chosen subset of cells that were active in the previous step [19].
- **SDR classifier**. Finally, the SDR classifier receives the set of activated cells from the temporal memory and predicts the likelihood of each possible input occurring the next step. This classifier is a fully connected neural network with a softmax activation function and cross-entropy loss function [9].

HTM has shown its efficiency and real-time capability of sequence prediction and anomaly detection [2, 12, 39]. It can learn both the temporal and spatial patterns using online training, unlike deep learning which uses batch offline-learning and cannot work with live streaming data. Moreover, HTM can adapt to changes in the data streams, which is expected to happen often in networks. All these properties make HTM the ideal candidate for the anomaly detection part of our problem.

3.3 Related Collaborative Machine Learning Methods

Conventional centralized machine learning (ML) approaches have achieved great success in many applications that have enough data storage and high computational power to train models. However, in distributed systems where multiple edge devices generate a large amount of data, the centralized learning is facing many challenges such as communication costs, reliability, and privacy and security concerns [1]. To address these issues, different collaborative distributed ML methods have been recently introduced to allow models to learn locally and benefit from other models, without the need to centrally store data. Federated Learning (FL) is a popular distributed ML approach developed by Google researchers, where models are trained on distributed devices under centralized control without sharing their local data [28]. The centralized unit aggregates all locally trained models parameters to form a global model shared by all edge devices. Split Learning is another distributed ML approach released by the MIT Labs to train deep learning network over multiple portions on distributed devices while mitigating the need to share data directly [18].

In this work, we build on the basic concept of FL to improve anomaly detection accuracy in a distributed environment. Conceptually, both FL and our approach leverage a number of distributed agents to improve model accuracy. Our approach, however, brings a novel addition to this form of learning. It shares models without sharing parameters (e.g., the weights and biases of a deep neural network) with the aim of maximizing the models efficiency. The main intuition underpinning our approach is the fact that different agents are exposed to different types and varying intensity of anomalies, which markedly impact their fitness as far as anomaly detection is concerned. Model sharing is essentially a sharing of expertise. Note that HTM lends itself naturally to such an approach, since its learning does not involve any form of parameter tuning.

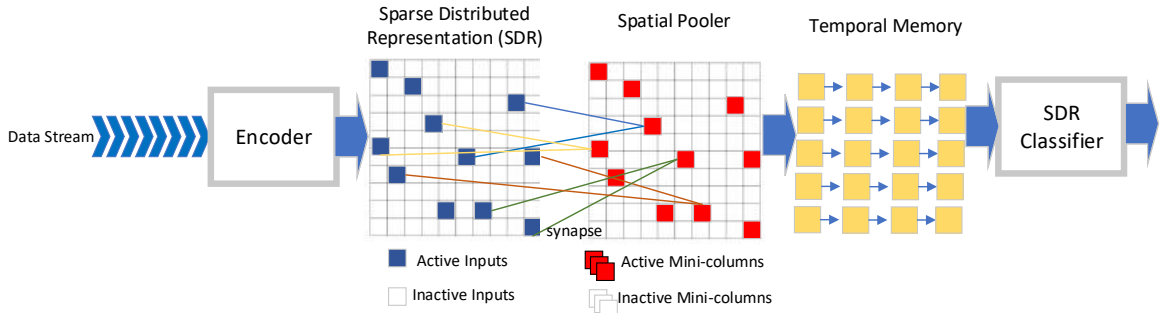


Figure 3: High-level architecture of the HTM system. The input data stream is encoded into SDR. This SDR is passed to spatial pooler. Finally, the temporal memory learns temporal sequences of these SDRs and makes predictions for future inputs.

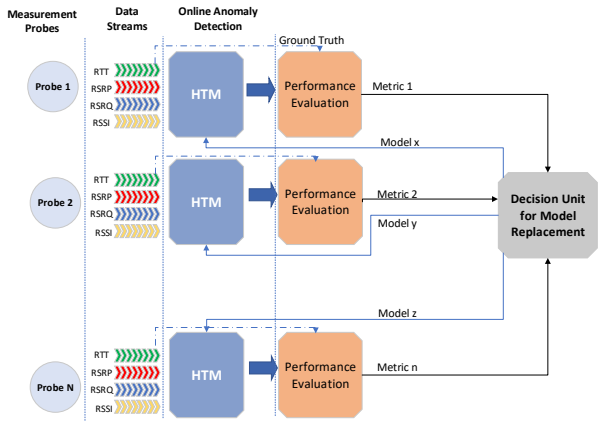


Figure 4: RCAD architecture.

4 DESIGN OF RCAD

Figure 4 depicts the architecture of RCAD. It comprises a set of probes that monitor various KPIs in real-time. These data streams are then fed to a local online anomaly detection module that involves an HTM and a performance evaluation component. The online anomaly detection modules then share their results with a module that decides on model replacement. Next, we elaborate on each of these modules.

4.1 Data Collection

We leverage monitoring streams of KPIs that are collected by a set of distributed probes. These probes are part of the Nornet Edge (NNE) platform. NNE is a measurement setup for measuring the performance and reliability of commercial mobile broadband networks, which consists of hundreds of stationary probes distributed across Norway [23]. Each probe is a single board computer that runs Linux and connects to commercial networks using commercial subscriptions. Here, we focus on four KPIs, which are the RTT between a probe and a central server that is part of NNE along with three KPIs that monitor the quality of radio connectivity and coverage. These are the received signal strength indicator (RSSI), the reference signal received power (RSRP) and the reference signal received quality (RSRQ). We have previously demonstrated that the contribution of RSSI, RSRQ and RSRP in predicting high delays in mobile networks [3]. For example, a worse RSRQ is indicative of a congested cell which leads to a high delay. The RTT is measured every second, while the other three KPIs are reported whenever there is a change. Our goal is to build a model that can accurately predict anomalies in RTT, i.e. increases in RTT. Each probe streams its KPIs measurements to the anomaly detection module in real-time.

4.2 Online Anomaly Detection

We use HTM for online anomaly detection. Using the architecture presented in Figure 3, we first feed the four KPIs data streams into scalar encoders. Besides, we use a datetime encoder to encode the corresponding timestamp value into time of day and day of week representations. We combine all these binary encodings using a multi encoder and pass them to a spatial pooler and a temporal memory. The temporal memory is always in learning mode, meaning that it learns from every sample it receives. The output from the temporal memory acts as the input to the anomaly detection algorithm in order to detect anomalies. The core of the HTM is not highly sensitive to parameters [2] [20], therefore we use the standard parameters set defined in [20]. We provide a description of how we configure HTM (i.e., the number of columns, the number of cells in each column and their connections) in Table 5 in the Appendix.

4.3 Decision Unit for Model Replacement

This module is responsible for managing the collaboration between the HTM models that are running at the probes. The model that experiences a large number of anomalies and throughout the time succeeds in detecting them, can share its spatial pooler and temporal memory with other models that have less exposure to anomalies or have low performance in detecting them. This allows each probe in the measurement infrastructure to participate in improving the performance of the overall system by sharing its experience on learning anomalies with other probes in the system. The decision unit is thus a centralized component that helps probes deciding on whether they need to replace their models by recommending using a model from another probe. In the following, we discuss two approaches for steering model replacement.

4.3.1 Model Replacement based on Scoring. The simplest approach is to pick the highest performing model at the moment of decision

and instruct all probes to adopt it. Note the performance of the model is assessed cumulatively from the start time to the exchange decision time. In this approach, we assume that the probe itself decides to replace its running model with another model, if its performance is below a certain threshold. For each probe, we count the number of failures and successes when detecting anomalies and pass them to the central decision unit for model replacement. The module decides that a certain probe should replace its running model, if the number of its failures exceeds the pre-defined threshold. The model with the highest success score is then chosen to replace the current one.

4.3.2 Model Replacement based on Deep Reinforcement Learning. Reinforcement learning (RL) is a learning paradigm that allows an agent to learn how to sequentially make decisions by interacting with an environment to maximize a certain reward. Formally, an RL task can be modelled as a Markov decision process (MDP) with state space S, action space A, initial state distribution $P0$, transition dynamics $P(s_{t+1}|s_t, a_t)$, and a reward function $r_{t+1} = r(s_t, a_t)$. The agent goal is to maximize the cumulative reward $R_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_{i+1}$, where $\gamma \in [0, 1]$ represents the discount factor that trades-off the immediate reward and future reward.

An agent's behavior is defined by a policy π, which maps a state to an action (deterministic policy) or maps a state to a probability distribution over all actions (stochastic policy). We can define a Q-function, which measures the expected accumulated rewards staring from any pair (s_t, a_t) and following the policy π, as shown below:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) | a_t \sim \pi(.|s_t), s_0 = s, a_0 = a] \tag{1}$$

The agent aims to find an optimal policy π^* when the Q^* function is maximized. In other words, a mapping from the environment states to actions, that maximizes the expected return. The optimality implies that Q^* satisfies the Bellman equation.

$$Q^*(s_t, a_t) = r_t + \gamma \mathbb{E}[max_{a'} Q^*(s_{t+1}, a^{'})] \tag{2}$$

Deep RL (DRL) [29] leverages the deep learning methods to work in complex environments, where the number of actions and states is large, or where the environment is non-deterministic. In such cases, evaluating every possible state-action pair to solve equation(2) becomes impossible.

Here, we formulate the model replacement problem in a distributed environment as an MDP. We aim to enhance the overall system performance through sharing learning experience between the probes. With the aid of an DRL agent, we can determine which model to choose and when to take the decision of the replacement. The agent is an ensemble of N models for N probes, which resides in the decision control unit. All models are running in real-time and produce anomaly predictions. The evaluation module produces the performance metrics of the predictions based on the ground truth for every pre-defined period τ. The agent collects these metrics as observations and takes decisions based on the accumulated reward. Next, we elaborate on the details of our formulation.

State Space. Here, we have N probes, each running a separate model that generates anomalies predictions. Every τ time, each model provides the agent (in the decision control) with performance indicators namely; true positives (TP) and false negatives (FN). These form the state space S, at time step t the state of the N probes (models) is given by:

$$s(t) = [TP_1(t), FN_1(t), TP_2(t), FN_2(t)..., TP_N(t), FN_N(t)] \tag{3}$$

Action Space. For each model n, we have N actions. Let A_n denote the action space for n model and $a_n \in A_n$ is the selected action for probe n. Therefore for N probes, we have $N \times N$, i.e., the number of actions. The agent has to take choose an action for each probe. At time step t, the selected action $a_t = [a_1, a_2, ...a_N]$, where $a \in [1..N]$ refers to which model to replace with. In case no replacement is recommended, the action value is set equal to the currently running model.

Reward Function. The reward function is defined to guide the agent to make desirable decisions in order to realize the objective of the system. Here, our objective is to maximize the overall performance of N models in detecting anomalies. As we are more interested in detecting high delays, we calculate the *Recall* based on the TP and FN from the state space for the overall system. The agent is rewarded, if the current recall is greater than the previous recall by a value of σ. We set $\sigma = 0.1$ to give emphasis to actions that considerably increase the system performance.

$$r(t+1) = \begin{cases} 1, & \text{if } \frac{Recall(t) - Recall(t-1)}{Recall(t-1)} \geq \sigma \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

DRL Agent. We implement our agent using the DQN algorithm, which use a neural network to approximate the optimal Q-value function $Q(s, a; \theta) \approx Q^*(s, a)$ [29]. Here, $Q(s, a; \theta)$ is called the Deep Q-network (DQN) and θ is the parameter of the neural network. The iterative update is used to train the Q-network and thus reduce the mean-squared error of the Bellman equation. The configuration of the DQN alongside the training process of the agent is explained in the Appendix.

5 EXPERIMENTS

In this section, we evaluate our proposed framework on latency measurements of 4G broadband network. We conduct extensive experiments to answer the following questions:

(1) Anomaly detection: does HTM outperform other state-of-the-art unsupervised methods for anomaly detection in mobile networks data?
(2) Robustness to noise: compared with the state-of-the-art methods, is HTM more robust to a high number of anomalies?
(3) Model replacement: in a distributed systems, can we utilize the nature of HTM to implement a collaborative learning paradigm that enhances the performance of anomaly detection?

5.1 Dataset

We leverage RTT measurements from one of the largest mobile operators in Norway. More specifically, we use per second RTT measurements from 10 probes. The length of each dataset is 15 days. Alongside the RTT measurements, we collect also RSSI, RSRP and RSRQ measurements. We define a threshold for classifying RTTs as

anomalous or not. This threshold is based on the RTTs distribution, where we consider the top 10% RTT values as anomalous, which is similar to the approach presented in [3]. Please refer to the appendix for more details about the dataset.

5.2 Evaluation Metrics

Precision (P), Recall (R), and F1-score (F1) are used to evaluate anomaly detection performance:

$$P = \frac{TP}{TP+FP}, \qquad R = \frac{TP}{TP+FN}, \qquad F1 = 2 * \frac{P*R}{P+R}$$

where TP is True Positives, FP is False Positives and FN is False Negatives.

We calculate the performance metric by comparing each predicted sample with the annotated ground truth, i.e point-wise evaluation. Other works such as [15, 38] use a point-adjust method that considers a window as anomalous as soon as one of the points it contains is detected as anomalous, even if the other points are not. Figure 1 shows that most anomalies are point anomalies. Furthermore, previous results have demonstrated that around 50% of the anomalies are point anomalies [3]. Therefore, in this context we use point-wise evaluation.

5.3 Results and Analysis

5.3.1 MV-HTM vs. other methods. To answer the first two research questions, we compare the effectiveness of multivariate HTM with four recent unsupervised anomaly detection methods. These methods are OmniAnomaly [38], USAD [4], NSIBF [15] and HTM+LSTM [12]. We choose these methods because they employ diverse approaches and have shown excellent performance.

Table 1 shows the prediction, recall and F1-score of these methods for datasets from three different probes (*Probe1, Probe3, Probe7*); each has a different delay profile (see Table 4 in Appendix). *Probe1*, *Probe3*, and *Probe7* have anomaly ratios of 23.23%, 41.95%, and 1% respectively. Recall that deep learning methods, i.e., OmniaAnomaly, USAD and NSIB are trained on normal data, while HTM uses online learning on mixed data (normal+ anomalous). Unlike OmniAnomaly and NSIBF, USAD allows to define a threshold for detecting anomalous samples. For OmniAnomaly and NSIBF, we tested possible anomaly thresholds and use the results linked to the highest F1-score.

From Table 1, we can see that multivariate HTM yields superior performance compared to the others across the three probes in terms of F1-score. Also, HTM+LSTM achieves good results as it uses HTM for prediction. However, the performance of the LSTM classifier reduces the overall performance of the framework.

We observe that most methods achieve good performance on *Probe3*. This is because it has a high rate of distinct anomalies i.e., anomalies that are associated with high magnitudes and last longer, which makes them easier to detect. From the deep learning methods, USAD has shown a consistent F-1 score for all three probes, but its precision drops significantly for *Probe3*. While NSIBF achieves the highest recall for *Probe1* and *Probe3*, it results in a very low precision, i.e. a very high extent of false alarms. This is mainly due to the fact that the above mentioned threshold is automatically defined by the method. In other words, this threshold was set to a value closer to normal samples. Moreover, we observe that the anomaly profile of the probes have a clear impact on the performance of OmniAnomaly and NSIBF. There is a notable performance degradation for *Probe 7*, which has fewer anomalies than the other two probes. The other probes experience almost similar performance to *Probe1, Probe3, Probe7* based on their anomalies level.

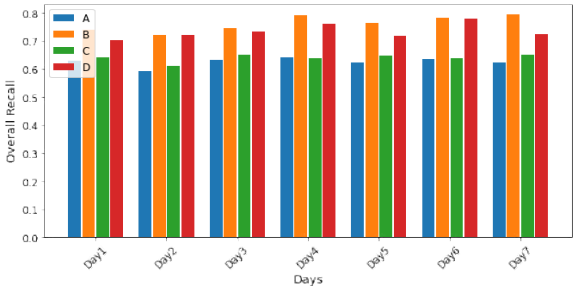


Figure 5: Overall system performance under four learning methods:(A) Without collaborative learning (B) Centralized learning (C) RCAD based on scoring (D) RCAD based on DRL.

5.3.2 Approaches to realizing RCAD. In this experiment, we compare four different learning schemes. The first involves only the anomaly detection part of RCAD that is implemented distributively *without collaborative learning*. The second is a *centralized learning* architecture, in which we combine the data streams from all probes in the system into one HTM model. In other words, we implement the anomaly detection part of RCAD in a centralized fashion. The third implements RCAD with model replacement based on *the scoring algorithm*. We use $threshold = 10$, which means the decision of the replacement is taken if the model fails in detecting 10 consecutive anomalies. The fourth implements RCAD with *DRL-based* model replacement. Here, we use $\tau = 120$ that the agent collects observations and takes actions of replacement every $2mins$. We evaluate these learning schemes using one week data and study the performance for both the overall system and individual probes. Figure 5 shows the overall recall for all probes in the system on daily basis. We can see that the centralized learning approach outperforms all other methods and increases the overall system recall by an average of 18% compared to models that learn individually. In some days, RCAD based on DRL achieves similar performance as the centralized learning. RCAD based on scoring shows slightly better performance than no collaborative learning. Table 2 shows the performance of individual probes (*Probe1, Probe3, Probe7*) under the different learning methods. Similar to the overall system performance in figure 5, the centralized learning outperforms all other methods in boosting probes performance. RCAD based on DRL comes second in improving the recall. Compared to the centralized learning, RCAD only improves the recall without a notable enhancement in the precision. This is due to the objective function that the DRL agent optimizes. Moreover, we can see that the probes with fewer anomalies do not improve their performance significantly in the RCAD with DRL-based model replacement, compared with other probes having medium or high anomaly ratios. This is owing to their low participation in the replacement process as shown in Figure 6. However, all probes benefit from the centralized learning regardless of the percentage of anomalies.

Table 1: Performance comparison between multivariate HTM method and state-of-the-art methods on mobile networks data.

Methods	Probe 1			Probe 3			Probe 7		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
OmniAnomaly [38]	0.621	0.658	0.639	0.698	0.704	0.701	0.378	0.122	0.184
USAD [4]	0.821	0.514	0.633	0.698	0.642	0.669	0.871	0.574	0.691
NSIBF [15]	0.323	0.785	0.458	0.616	0.755	0.679	0.064	0.091	0.0754
HTM+LSTM [12]	0.708	0.613	0.657	0.651	0.647	0.650	0.729	0.584	0.647
MV HTM	**0.732**	**0.697**	**0.714**	**0.719**	**0.727**	**0.721**	**0.782**	**0.632**	**0.699**

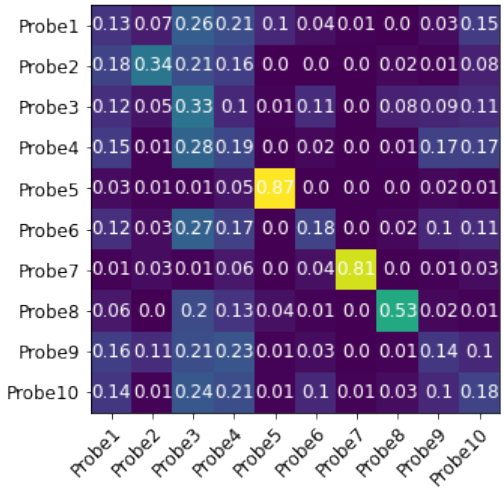


Figure 6: Participation of each probe to other probes in model replacement decisions taken by the DRL agent. The value of $cell_{ij}$ refers to the fraction of replacement decisions taken by $Probe_i$ to use $Probe_j$ from all decisions.

Further, we extend our experiments for model exchange DRL-based to understand the optimal policy for replacement decisions. In this experiment, we test our trained agent on one day data and check the actions of replacement (i.e., 720 actions). Figure 6 shows the participation of each probe in the replacement decisions through this snapshot. We can see that probes with fewer anomalies (for example *Probe5, Probe7, Probe8*) do not make the most of this collaborative scheme and they stay on their current running models without replacement with other probes. Moreover, other probes are less likely to replace their running models with models of low anomalies probes. On the other hand, probes with medium number of anomalies (for example *Probe1, Probe9, Probe10*) are replacing their models with the ones from high anomalies probes (for example *Probe3*) most of the time with participation factor of 0.26, 0.21, 0.24, respectively. In summary, probes with a high rate of anomalies contribute the most in this collaborative method, which confirms the intuition behind model replacement.

5.3.3 Effect of parameters. Having seen that RCAD yields excellent performance, we now turn to investigating its sensitivity to parameters values. The first parameter we look at is the threshold for deciding to replace the current model by a probe. This threshold refers to the number of consecutive failures in detecting anomalies. We test three different values, namely 5, 10 and 15 which result in an overall recall of 0.649,0.641 and 0.633, respectively for one-day long dataset, i.e. the first day in our dataset. Note that we have set this threshold to 10 in our evaluations. We observe minimal differences between the three values. A lower threshold value seems to result in a slightly better accuracy which comes at the cost of a higher communication overhead though.

The second parameter we study is τ, which defines how frequently the DRL agent collects the performance data from probes to decide model replacement. Table 3 shows the agent's behavior during training for different values of τ using the same setup of hyperparameters. Note that we test on the same dataset as for the threshold parameter above. While a higher value of τ leads to a faster convergence due to the reduction in training samples, it does result in a marginally lower overall recall.

In summary, RCAD exhibits marginal sensitivity to parameterization. This allows for adjusting parameters to choose a desired balance between accuracy and overhead.

Table 3: Trade-off between training time and the system performance under different values of τ.

Value of τ (mins)	Convergence (episodes)	Overall recall
2	900	0.703
15	700	0.683
20	650	0.677

6 CONCLUSION

In this paper, we propose RCAD, a Real-time Collaborative Anomaly Detection framework for multivariate time series based on HTM and model replacement. The HTM model predicts anomalies in real-time by learning sequences in data streams. Model replacement allows probes with poor performing models to benefit from other probes that have better performing models. We evaluate RCAD on a two-week long dataset from 10 probes that are part of a setup for measuring the performance of commercial mobile broadband networks in Norway. RCAD has demonstrated superior performance to four state-of-art methods in terms of F1-score. Moreover, it has demonstrated robustness to anomalies ratio and stability across probes with different anomaly profiles. RCAD with DRL-based model replacement has achieved a level of performance comparable to a centralized HTM. For future work, it would be interesting to test our method on applications with similar requirements as our networking use case in addition to looking deeper into different model exchange methods.

Table 2: Performance comparison between RCAD and centralized learning.

Methods	Probe 1			Probe 3			Probe 7		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
No CL	0.732	0.697	0.714	0.719	0.727	0.721	0.782	0.632	0.699
Centralized	0.792	0.735	0.762	0.741	0.731	0.736	0.831	0.714	0.768
RCAD-Scoring	0.741	0.707	0.723	0.725	0.718	0.721	0.793	0.642	0.710
RCAD-DRL	0.745	0.726	0.735	0.731	0.729	0.730	0.791	0.686	0.735

REFERENCES

[1] Haftay Gebreslasie Abreha, Mohammad Hayajneh, and Mohamed Adel Serhani. 2022. Federated Learning in Edge Computing: A Systematic Survey. *Sensors* 22, 2 (2022), 450.

[2] Subutai Ahmad, Alexander Lavin, Scott Purdy, and Zuha Agha. 2017. Unsupervised real-time anomaly detection for streaming data. *Neurocomputing* 262 (2017), 134–147.

[3] Azza H Ahmed, Steven Hicks, Michael Alexander Riegler, and Ahmed Elmokashfi. 2021. Predicting High Delays in Mobile Broadband Networks. *IEEE Access* 9 (2021), 168999–169013.

[4] Julien Audibert, Pietro Michiardi, Frédéric Guyard, Sébastien Marti, and Maria A Zuluaga. 2020. USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3395–3404.

[5] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data mining and knowledge discovery* 31, 3 (2017), 606–660.

[6] Raghavendra Chalapathy and Sanjay Chawla. 2019. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407* (2019).

[7] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41, 3 (2009), 1–58.

[8] Tesnim Charrad, Kaouther Nouira, and Ahmed Ferchichi. 2019. Use of Hierarchical Temporal Memory Algorithm in Heart Attack Detection. *International Journal of Computer and Systems Engineering* 13, 5 (2019), 308–311.

[9] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. 2016. Continuous online sequence learning with an unsupervised neural network model. *Neural computation* 28, 11 (2016), 2474–2504.

[10] Yuwei Cui, Subutai Ahmad, and Jeff Hawkins. 2017. The HTM spatial pooler—A neocortical algorithm for online sparse distributed coding. *Frontiers in computational neuroscience* 11 (2017), 111.

[11] Nan Ding, Huanbo Gao, Hongyu Bu, and Haoxuan Ma. 2018. RADM: real-time anomaly detection in multivariate time series based on Bayesian network. In *2018 IEEE International Conference on Smart Internet of Things (SmartIoT)*. IEEE, 129–134.

[12] Noha Ossama El-Ganainy, Ilangko Balasingham, Per Steinar Halvorsen, and Leiv Arne Rosseland. 2020. A New Real Time Clinical Decision Support System Using Machine Learning for Critical Care Units. *IEEE Access* 8 (2020), 185676–185687.

[13] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for Time Series Classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[14] Nick Feamster and Jennifer Rexford. 2017. Why (and how) networks should run themselves. *arXiv preprint arXiv:1710.11583* (2017).

[15] Cheng Feng and Pengwei Tian. 2021. Time series anomaly detection for cyber-physical systems via neural system identification and bayesian filtering. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 2858–2867.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27 (2014).

[17] Michael Gundall, Mathias Strufe, Hans D Schotten, Peter Rost, Christian Markwart, Rolf Blunk, Arne Neumann, Jan Grießbach, Markus Aleksy, and Dirk Wübben. 2021. Introduction of a 5G-enabled architecture for the realization of industry 4.0 use cases. *IEEE access* 9 (2021), 25508–25521.

[18] Otkrist Gupta and Ramesh Raskar. 2018. Distributed learning of deep neural network over multiple agents. *Journal of Network and Computer Applications* 116 (2018), 1–8.

[19] Jeff Hawkins and Subutai Ahmad. 2016. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits* 10 (2016), 23.

[20] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin. 2016. Biological and Machine Intelligence (BAMI). (2016). https://numenta.com/resources/biological-and-machine-intelligence/ Initial online release 0.4.

[21] Donald Olding Hebb. 2005. *The organization of behavior: A neuropsychological theory*. Psychology Press.

[22] Anand Padmanabha Iyer, Li Erran Li, and Ion Stoica. 2017. Automating diagnosis of cellular radio access network problems. In *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking*. 79–87.

[23] Amund Kvalbein, Džiugas Baltrūnas, Kristian Evensen, Jie Xiang, Ahmed Elmokashfi, and Simone Ferlin-Oliveira. 2014. The Nornet Edge Platform for Mobile Broadband Measurements. *Computer Networks* 61 (2014), 88–101.

[24] Mads Lauridsen, Lucas Chavarria Gimenez, Ignacio Rodriguez, Troels B Sorensen, and Preben Mogensen. 2017. From LTE to 5G for connected mobility. *IEEE Communications Magazine* 55, 3 (2017), 156–162.

[25] Dan Li, Dacheng Chen, Baihong Jin, Lei Shi, Jonathan Goh, and See-Kiong Ng. 2019. MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks. In *International Conference on Artificial Neural Networks*. Springer, 703–716.

[26] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation forest. In *2008 eighth ieee international conference on data mining*. IEEE, 413–422.

[27] Benjamin Lucas, Ahmed Shifaz, Charlotte Pelletier, Lachlan O'Neill, Nayyar Zaidi, Bart Goethals, François Petitjean, and Geoffrey I Webb. 2019. Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery* 33, 3 (2019), 607–635.

[28] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

[30] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.

[31] Guansong Pang, Chunhua Shen, Longbing Cao, and Anton Van Den Hengel. 2021. Deep learning for anomaly detection: A review. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–38.

[32] Daehyung Park, Yuuna Hoshi, and Charles C Kemp. 2018. A multimodal anomaly detector for robot-assisted feeding using an LSTM-based variational autoencoder. *IEEE Robotics and Automation Letters* 3, 3 (2018), 1544–1551.

[33] Georgios Patounas, Xenofon Foukas, Ahmed Elmokashfi, and Mahesh K Marina. 2020. Characterization and Identification of Cloudified Mobile Network Performance Bottlenecks. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 2567–2583.

[34] Cisco public. 2019. *Cisco Visual Networking Index: Forecast and Trends, 2017–2022*. Retrieved Feb 8, 2022 from https://twiki.cern.ch/twiki/pub/HEPIX/TechwatchNetwork/HtwNetworkDocuments/white-paper-c11-741490.pdf

[35] Scott Purdy. 2016. Encoding data for HTM systems. *arXiv preprint arXiv:1602.05925* (2016).

[36] Mayu Sakurada and Takehisa Yairi. 2014. Anomaly detection using autoencoders with nonlinear dimensionality reduction. In *Proceedings of the MLSDA 2014 2nd workshop on machine learning for sensory data analysis*. 4–11.

[37] Konstantinos Samdanis and Tarik Taleb. 2020. The road beyond 5G: A vision and insight of the key technologies. *IEEE Network* 34, 2 (2020), 135–141.

[38] Ya Su, Youjian Zhao, Chenhao Niu, Rong Liu, Wei Sun, and Dan Pei. 2019. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2828–2837.

[39] Chundong Wang, Zhentang Zhao, Liangyi Gong, Likun Zhu, Zheli Liu, and Xiaochun Cheng. 2018. A distributed anomaly detection system for in-vehicle network using HTM. *IEEE Access* 6 (2018), 9091–9098.

[40] Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. 2018. Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In *International conference on learning representations*.

A SUPPLEMENTRAY MATERIAL FOR REPRODUCIBILITY

We publish our dataset and code in a GitHub repository[2] for public to reproduce our experiments.

A.1 Datasets

In this work, we use measurements for RTT, RSRP, RSRQ from 10 probes recorded every second for two weeks. Each dataset has training and testing subsets. Anomalies are presented in testing dataset. However, for HTM experiments the training datasets contain anomalies as well. Anomalies are being labelled based on the data distribution considering the top %10 as anomalies [3]. Table 4 shows the statistics for these datasets.

Table 4: Dataset Information.

Dataset name	Training Set of Normal data size	Testing set size	Anomaly ratio
Probe 1	479654	580157	23.23%
Probe 2	495265	543873	12.34%
Probe 3	416544	578450	41.95%
Probe 4	392363	573953	34.88%
Probe 5	520378	580028	8.61%
Probe 6	553435	542783	14.16%
Probe 7	526464	481394	1%
Probe 8	483626	532822	12.10%
Probe 9	496907	570786	22.63%
Probe 10	400416	567403	26.72%

A.2 Hierarchical Temporal Memory (HTM) Implementation

For the implementation of HTM model, we use the Numenta Platform for Intelligent Computing (NuPIC) published by Numenta on Github.[3]. Noting that this library is only compatible with Python 2.7.

Model parameters are defined in a file which contain many details about how the HTM network will be constructed, what encoder configurations will be used, and spatial pooler and temporal memory parameters. The model parameters we used in this experiment are shown in Table 5.

Table 5: HTM Model Parameters.

Parameter Name	Value
Number of columns	2048
Number of cells per column	32
Activation Threshold	8
Initial Permanence	0.21
Connected Permanence	0.5
Permanence Increment	0.1
Permanence Decrement	0.1
Max Segments Per Cell	255

[2]https://github.com/azza8903/HTM-MODEL_EXCHANGE/
[3]https://github.com/numenta/nupic

A.3 Reinforcement Learning Agent Implementation for Model Exchange

To implement DQN agent, we use a deep neural network of one hidden layer. We use Rectified Linear Unit (ReLU) as an activation function for the hidden layer. The input layer size of the Q-network represents the state space size which is 20 (number of probes in the system $(N) = 10$). The output layer size is 100 which represents the total number of actions. We trained the agent using same algorithm in [30]. The training parameters of the DQN are listed in Table 6. We develop DQN agent by using PyTorch(version 1.8.1)[4] which is compatible with Python2.7.

Table 6: DQN Training Hyper-parameters.

Parameter	Value
Experience-replay memory size	40000
Batch size	64
Target network update frequency B	1000
Learning rate	0.00025
Loss function	MSE
Optimizer	Adam Optimizer

[4]https://pytorch.org/

Article III

Ahmed, A. H., and Elmokashfi, A. **ICRAN: Intelligent Control for Self-Driving RAN Based on Deep Reinforcement Learning**, in IEEE Transactions on Network and Service Management, vol. 19, no. 3, pp. 2751-2766, Sept. 2022,
DOI: **https://doi.org/10.1109/TNSM.2022.3191746**

ICRAN: Intelligent Control for Self-Driving RAN Based on Deep Reinforcement Learning

Azza H. Ahmed, *Member, IEEE*, and Ahmed Elmokashfi

***Abstract*—Mobile networks are increasingly expected to support use cases with diverse performance expectations at a very high level of reliability. These expectations imply the need for approaches that timely detect and correct performance problems. However, current approaches often focus on optimizing a single performance metric. Here, we aim to address this gap by proposing a novel control framework that maximizes radio resources utilization and minimizes performance degradation in the most challenging part of cellular architecture that is the radio access network (RAN). We devise a method called Intelligent Control for Self-driving RAN (ICRAN) which involves two deep reinforcement learning based approaches that control the RAN in a centralized and a distributed way, respectively. ICRAN defines a dual-objective optimization goals that are achieved through a set of diverse control actions. Using extensive discrete event simulations, we confirm that ICRAN succeeds in achieving its design goals, showing a greater edge over competing approaches. We believe that ICRAN is implementable and can serve as an important point on the way to realizing self-driving mobile networks.**

***Index Terms*—Self-driving network, slicing, RAN, resource allocation, performance optimization, deep reinforcement learning, ns-3 simulation, DDPG.**

I. Introduction

THE FIFTH generation mobile network, 5G, has transformed the mobile network into a multi-service architecture that supports diverse use cases with varying requirements [1]. 5G virtualizes network resources and chains them into end-to-end network slices that are adapted to use cases' requirements. This flexibility makes mobile networks increasingly complex to manage [2]. Furthermore, several envisioned use cases like public safety communication and industrial control have stringent performance expectations. Hence, multi-slice 5G and 6G networks must be capable of quickly detecting and correcting performance degradation. Current mobile networks resort to pre-configured priorities, over-provisioning and at best implementing traditional closed loop control systems with a limited scope like in the case of self-organizing networks (SON) [3]. The need for intelligent automation has motivated the academia and industry to argue for building "autonomous" or "self-driving" networks, where network management and control decisions are made in real-time and in an automated fashion [4]. For instance, the Open Radio Access Network (O-RAN) architecture, which aims to realize the RAN as a set of visual network functions on commodity hardware, has identified supporting intelligent RAN control as a key design goal [5]. Despite all these effort, building "self-driving" mobile networks that are practically deployable has largely remained unrealized.

A major challenge in this respect, having in mind the scale of mobile networks, involves devising a control architecture that balances complexity and overhead. Further, there is a lack of unified control approaches that can deliver on multiple objectives (e.g., maximize resource utilization while ensuring an acceptable level of performance). More specifically, the existing proposals focused on tracking and optimizing a single metric like coverage [6], power management [7], throughput [8] and resource sharing [9] at a time. It flows directly from this limited focus that current approaches resort to often applying a single control action, e.g., adjusting base station transmit power. However, a multi-slice network is by definition a multi-service network. Hence, tracking a single metric is bound to assure quality for a subset of the services that run on the network. A viable approach to realizing self-driving mobile networks must be able assure quality for all running slices according to their priority and service level agreements (SLAs). Achieving this requires choosing and mixing diverse control actions, e.g., simultaneously adjusting coverage and optimizing resource allocation. Here, we aim to bridge this gap by proposing a machine learning based approach to manage resource utilization and performance in the RAN. We focus on the RAN, because its performance and reliability heavily impact users' quality of experience [10]. Our approach tracks and optimizes diverse use cases. To this end, it employs several different control actions.

We propose, ICRAN, an intelligent control scheme for a multi-slice RAN. ICRAN leverages deep reinforcement learning (DRL) to derive strategies for maximizing resource utilization and minimizing SLA violations under different network conditions. Reinforcement learning is a machine learning (ML) approach where intelligent agent/agents interpret and interact with their environment and learn how to achieve certain objectives via cycles of trial and error based learning. We introduce both a centralized and distributed control schemes. In the former, a single controller optimizes the configurations of all base stations, while each base station has own controller in the

Manuscript received 4 May 2022; revised 4 July 2022; accepted 13 July 2022. Date of publication 18 July 2022; date of current version 12 October 2022. The associate editor coordinating the review of this article and approving it for publication was M. Tornatore. *(Corresponding author: Azza H. Ahmed.)*

Azza H. Ahmed is with the Center for Resilient Networks and Applications, Simula Metropolitan Center for Digital Engineering, 0167 Oslo, Norway, and also with the Center for Resilient Networks and Applications, Oslo Metropolitan University, 0176 Oslo, Norway (e-mail: azza@simula.no).

Ahmed Elmokashfi is with the Center for Resilient Networks and Applications, Oslo Metropolitan University, 0176 Oslo, Norway.

Digital Object Identifier 10.1109/TNSM.2022.3191746

latter. Following an initial training phase, the controller continuously monitors the state of the network and immediately reacts to performance degradation by executing actions that extend coverage and regulate resource usage. We implement ICRAN using the OpenAI Gym framework [11] and the ns-3 simulator [12]. Our evaluations show that ICRAN converges quickly to strategies that help maximizing radio resource utilization and minimizing SLA violations for the entire network. ICRAN outperforms approaches that leverage DRL, implement adaptive priority-based resource management as well as those resorting to heuristics to react to network changes. The benefit from ICRAN spans regimes where the network is lightly loaded, running at its capacity, and is heavily loaded. For example, ICRAN utilizes 97% of available radio resources when the network is loaded at 200% that is 7% higher than the next best method. At the same time, it reduces the number of SLA violations for slices with stringent requirements, that is achieved by the next best method, by up to a factor of three. This work is the first to apply deep reinforcement learning to collectively solving multiple RAN control problems. The previous work has mainly focused on optimizing for a single control problem.

A. Contributions

The main contributions of this paper are summarized in the following:

1) We propose ICRAN a novel control framework based on DRL that is capable of maximizing resource utilization and minimizing SLA violations in a multi-slice RAN. ICRAN introduces a novel reward function and an action space with diverse actions which allows for optimizing multiple objectives collectively, namely antenna tilt, traffic load balancing, and resource allocation.
2) We investigate two different control architectures for our framework; centralized and distributed control. In the centralized framework, we formulate the problem as a single-agent DRL whereas, the distributed problem is solved via a multi-agent DRL.
3) Finally, we validate the performance of our proposed framework through extensive simulations using ns-3. The experiment results show that our method outperforms the state-of-the-art methods in radio resources management. Moreover, the advantage of our method persists under different networking conditions such as high congestion and radio failure.

B. Paper Structure

The remainder of the paper is organized as follows. The next section gives the background of this work, while Section III covers related research on RAN slicing and the application of DRL in mobile networks. In Section IV, we discuss the overall system model from a high-level perspective. We describe in Section V the scheduling algorithm with slicing constraints that we consider. Then, we proceed to elaborate our DRL problem formulation and the algorithms we develop to solve the problem. We present our experimental setup in Section VI and we evaluate the performance of ICRAN and compare it with some baselines and state-of-the-art methods in Section VII. In Section VIII, we discuss our findings, the implementation considerations and the limitations of this work and how to address them in the future. Finally, a brief concluding remarks are provided in Section IX.

II. Background

Unlike the previous generations of mobile networks, 5G is designed to support a wide range of services with diverse requirements. These requirements span a wide range in terms of expected throughout, latency, reliability, mobility and number of devices [13]. A single static network architecture, however, can not cater for this diversity. This motivated a pivot towards realizing multiple isolated logical networks, with different configurations, over the same physical infrastructure, i.e., network slicing [14], [15], [16], [17]. Recent advances in defining network elements and network configurations in software have made this pivot possible. Key technologies in this respect are software defined networking (SDN), network function virtualization (NFV) and cloud-native network functions (CNFs). SDN decouples the network data and control planes and centralizes network control which allows for a full network programmability [18]. NFV transfers network functions like routers and firewalls from specialized physical implementations (i.e., a hardware box with tailored software) to software implementations that can run as virtual machines on a general purpose computers. VNFs can be chained to form an end-to-end network architecture [19]. Finally, CNF is a natural evolution of VNF that shrinks them to run as containers and optimizes them to run in the cloud [20]. SDN and NFV/CNF complement each other towards building an end-to-end software defined and programmable network architectures. In the Long-Term Evolution (LTE) mobile core network, the data and control plane functions are realized by dedicated hardware that implements each specialized function. However, the 5G core is designed to be "cloud-native", in the sense that the functions that handle the control and data planes, e.g., UPF and AMF, could be deployed as VNFs/containers on a cloud infrastructure. The use of NFV and SDN has been started from core and networking middleboxes and extended to RAN functions. The development of open and intelligent RAN (O-RAN) has received great attention [21]. O-RAN is proposed to enhance the RAN performance through virtualized network elements and open interfaces that incorporate intelligence into the RAN. O-RAN introduces programmable components that can run optimization routines with closed-loop control and orchestrate the RAN. Specifically, the O-RAN has logical controllers that monitor the status of the network (e.g., number of users, load, throughput, resource utilization) and process this data leveraging AI/ML algorithms to determine and apply control policies and actions on the RAN, for example, network and RAN slicing, load balancing, handovers and scheduling [5].

III. Related Work

A. RAN Slicing

End-to-end slicing involves virtualizing the RAN, the transport and the core. The former is specific to mobile networks

while the latter two are common across different types of networks, e.g., data centers. Further, RAN slicing requires an efficient approach to resource management given the limited nature of the virtualized-resource in question, i.e., radio spectrum. To this end, there are several proposals which we discuss next.

Various traditional algorithms have been proposed for tackling the resource scheduling and allocation problems in the RAN. Nojima *et al.* [22] presented three resource allocation methods for RAN slicing by slightly modifying the conventional MAC scheduling algorithms: 1) a static allocation method, which allocates Resource Blocks (RBs), that is the smallest units of resources in frequency and time that can be allocated to a user, in a fixed manner regardless of the channel conditions of users in each slice, 2) a round-robin allocation algorithm, which allocates RBs to each slice sequentially and based on the channel conditions of users in each slice, 3) a per-user priority algorithm that allocates RBs to users based on their priority within their respective slice. Their experiments showed that the priority-based algorithm achieved higher throughput compared to that resulted by the static and round-robin algorithms. However, all these methods do not consider satisfaction of the slice requirements when allocating RBs to a slice. Thus, Shrivastava *et al.* [23] proposed a method that allocates RBs to slices taking into account the desired SLA. This approach allows for a flexible assignment of RBs, which allocates temporarily unused RBs to slices in need with the possibility of allocating more RBs than necessary. To address this, Bakri *et al.* [24] proposed a data-driven mechanism for sharing RAN physical resources among different network slices based on optimal value for RBs. The proposed algorithm calculates the optimal value for the radio resources based on the Channel Quality Indicator (CQI) reports collected by the base stations. The algorithm is running at the slice orchestrator level which adapts to two slices with different quality of service (QoS) requirements, specifically ultra reliable and low latency communications (URLLC) slice and enhanced Mobile Broadband (eMBB) slice. To reduce the overhead of sending the CQI values between the base station and the slice orchestrator, the authors presented a machine learning model to predict the user equipment (UE) channel stability. If the channel quality is relatively stable, the CQI values do not vary much in time. Therefore, frequent CQI reports will not affect the performance of the slicing algorithm. Unlike our work, the proposed method calculates the required radio resources based on only one key performance indicator (KPI), i.e., either latency or throughput and thus cannot directly support other types of slices. Moreover, their results showed that there is a threshold on the number of users in each slice when the SLA is violated without considering how to minimize those SLA violations.

Aligned with the recent advances in DRL, Mei *et al.* [8] presented a hierarchical framework based on integrating the deep deterministic policy gradient (DDPG) and double deep-Q-network algorithm to solve RAN slicing problem. Specifically, this framework consists of two controllers: an upper-level controller which adjusts the slice configuration to improve QoS performance at a coarse granularity and a lower-level controller that schedules network resources and power allocation to active UEs in each network slice at a fine granularity. Their results showed that RBs in RAN slices can be managed efficiently using the DDPG for RB allocation. However, in this method, if the number of slices is different from the number of slices during training, RB allocation to slices is impossible. Therefore, RB allocation independent of the number of slices was proposed in [25], where Liu *et al.* presented a method called DeepSlicing which tackles the problem of resource allocation in multi-slicing networks in two stages. The first stage allocates resources to users within a slice through DRL that learns the optimal policy in each network slice to maximize the overall utilities of users in the slice while satisfying the users' SLA. The second stage coordinates the resource allocation across the network slices. Similar to our work, this work leverages the DRL advances in RAN resource management. However, our work goes beyond the efficient resources allocation solution to explore a large action space containing other possible actions such as antenna tilt optimization and load balancing between the base station to achieve near-zero violations of slices' SLA. To address the problem of the long training time needed by DRL methods, Abouaomar *et al.* [26] proposed a federated DRL mechanism to collaboratively train a DRL model for bandwidth allocation in RAN slicing. Their simulation results have shown that the model trained using federated learning is more robust against environment changes compared to models trained separately by each mobile virtual network operator.

B. Deep Reinforcement Learning in Mobile Networks

The advances in DRL have led to outstanding success in various domains. DRL has been recently proposed for solving many wireless communication problems. Several surveys summarized these works. For example, Luong *et al.* [27] provided a comprehensive overview of deep reinforcement learning application in communications and networking such as dynamic network access, data rate control, wireless caching, data offloading, network security and connectivity preservation. Compared to [27] which focused on single agent problems, Feriani and Hossain [28] presented an overview of both single-agent and multi-agent reinforcement learning as key enabling technologies of future wireless networks. They highlighted the potential for applying cooperative multi-agent reinforcement learning to different domains such as mobile edge computing (MEC), unmanned aerial vehicles (UAV) networks and massive MIMO. Other surveys reviewed the application of deep reinforcement learning algorithms in specific domains such as Internet of Things (IoT) [29], URLLC in 6G networks [30], vehicular networks in 6G [31] and mobile edge caching [32]. Recently, Seid *et al.* [33] proposed a multi-UAV enabled IoT edge approach for dynamic task offloading and resource allocation leveraging multi-agent DRL methods. They aimed to minimize the overall network computation cost while ensuring the QoS requirements of IoT devices or UEs in the IoT network.

In general, the previous proposals can be divided into two groups based on the action space. The first applies deep

Q-learning to problems with a discrete action space, while the second applies actor-critic methods to problems with a continuous action space. Further, almost all existing work revolves around executing a single principal action. For example, Nasir and Guo [7] proposed a power allocation scheme based on deep Q-learning model (DQN). Each transmitter collects channel state information and QoS information from several neighbors and adapts its own transmit power accordingly. Also, Li *et al.* [34] used DQN method to tackle the resource allocation multi-user computation offloading in wireless MEC. Recently, Ren *et al.* [35] addressed the problem of dynamic resource allocation for MEC slicing system using DDPG algorithm. They formulated the resource allocation problem as Markov decision process (MDP) in which, the wireless resources and computing resources are configured dynamically according to the requirements of different types of slices to maximize the network operator revenue. Similarly, Seid *et al.* [36] leveraged DDPG algorithm to optimize the computational costs and resource allocation to satisfy the QoS of Edge IoT devices. They proposed a collaborative framework where each agent learns from the previous offloading experiences and dynamically associates the nearest computational node with the UAV network. We refer the reader to the aforementioned surveys and the references for a comprehensive overview of previous efforts.

In this paper, we propose ICRAN as a control approach for performance optimization based on DRL in a multi-slice RAN. We present two variants of ICRAN: single agent and multi-agent. ICRAN uses an actor-critic method due to its continuous action space. This work is, to the best of our knowledge, the first work to date that presents a DRL-based approach that both maximizes radio resource utilization and minimizes SLA violations simultaneously. Also, our work is the first, in this application area, to propose the use of three inherently different categories of actions.

IV. System Model

In this section, we present the problem statement, the system and traffic models, and our control approach.

A. Problem Definition and System Architecture

Network performance optimization represents one of the major challenges for mobile network operators, especially in RAN [37]. Near-future mobile networks are going to support users with different service requirements by leveraging network slicing. Therefore, an efficient scheduling mechanism is essential for allocating available resources to these different types of services. Since traffic is dynamic, a static resource allocation is bound to fall short in maximizing resource utilization while maintaining the SLA. Hence, there is a need for an adaptive control mechanism that steers the network in response to traffic demand variations and anomalies. This paper proposes a DRL-based framework, *ICRAN*, that maximizes resource utilization and minimizes SLA violations in a multi-slice RAN.

We investigate RAN control in LTE cellular network consisting of M eNBs that serve N users, who connect to

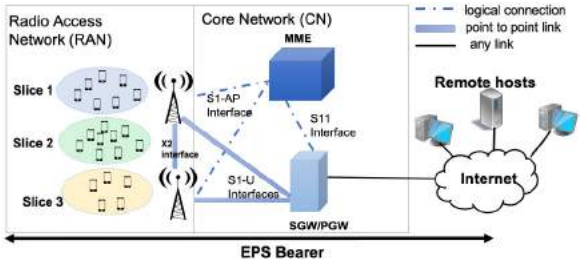


Fig. 1. The LTE reference architecture that we use in our framework.

three different slices with diverse service requirements (see Figure 1). The number of users that connect to the three slices is N_{S1}, N_{S2} and N_{S3}, respectively. The core network contains the serving gateway (SGW), packet data network (PDN) gateway (PGW), and the mobility management entity (MME). Among other functions, the MME is responsible for paging procedures of UEs upon arrival. When a new UE arrives to the network, a logical channel is established between the UE and eNB called a radio bearer, which also connects the UE to IP-based networks through the evolved packet core (EPC). The radio bearer is associated with QoS parameters based on the application it serves. A key parameter in the respect is the QoS Class Identifier (QCI).

We choose to focus on LTE because realistic 5G simulation models are still under development. For example, the current models do not support handovers yet [38], which is an essential feature in mobile networks and in our framework. Nevertheless, we believe that this choice has no impact on the generalizability of our results to 5G networks, because we are not making any assumption that is only limited to LTE. Our simulation implements all the 4G core network components (PGW, SGW, PDN, MME and EPC), UE, MAC layer, as well as all higher protocol such as Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP), Radio Resource Control (RRC) which closely resemble that of 5G. Unlike 4G, 5G base stations (i.e., gNBs) and UEs might support multiple numerologies, however this feature does not affect the functionality of ICRAN.

In our system model, we assume a DownLink (DL) dominated traffic model, in agreement with today's traffic patterns [39]. Generally speaking, LTE supports two traffic patterns:

- Guaranteed Bit-Rate (GBR): GBR traffic requires a constant throughput, irrespective of the required resources it takes to fulfill it. Voice-over-IP (VoIP) is an example of an application that expects a GBR.
- Non-guaranteed Bit-Rate (Non-GBR): this traffic pattern does not have rigid throughput requirements and can use any unused resources left by guaranteed services. Applications like Web browsing can be considered as a non-GBR service. Based on the users' channel conditions and scheduler decision, the throughput is determined.

LTE introduced several QCIs to tag different traffic patterns, which have been used by various works to realize the slicing [40], [41], [42]. Here, we leverage QCI values to define three slices that we summarize in Table I. The first slice represents the GBR service which has the strictest SLA both in

TABLE I
SLICES DEFINITION

Slice	Application	QCI	Resource Type	Priority Level	Delay Budget	Packet Loss Rate	Minimum UE Throughput(Kbps)
1	VoIP	1	GBR	2	$100ms$	$10.E-2$	64
2	Video	7	non-GBR	6	$300ms$	$10.E-6$	525
3	Best Effort	9	non-GBR	9	$300ms$	$10.E-6$	N/A

terms of throughput and delay requirements. Both *Slice 2* and *Slice 3* are non-GBR services, however, *Slice 2* has a minimum throughput that the network should provide. Finally, *Slice 3* is the best-effort slice that uses the remaining resources. We assume that every user is a member of only one of the three slices. This is a reasonable assumption for the majority of users.

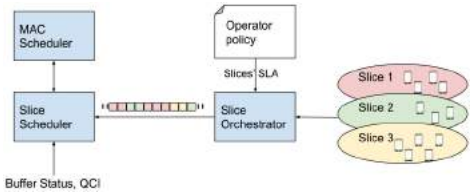


Fig. 2. Our RAN slicing model.

B. Control Frameworks and Levers

We propose two different control architectures for ICRAN.

1) Centralized ICRAN (ICRAN-C): In this architecture, we consider a single centralized controller (agent) that can fully observe the network information and reconfigure the entire network accordingly. Even though a centralized decision-making may be the best in performance, it is usually impractical due to the potential signaling overhead. Furthermore, in practical implementations, a centralized solution can be slow. It is difficult for a centralized algorithm to quickly find an optimal solution because the search space increases exponentially as the size of the network increases [43].

2) Distributed ICRAN (ICRAN-D): To avoid the pitfalls of centralized control, a partially centralized or a fully decentralized control architecture is needed. Here, each eNB takes its own decisions without considering other eNBs' state and decision, or with little information about all or some eNBs, for example neighbouring eNBs. Having a fully independent decentralized architecture, may result in conflicting strategies as each eNB is essentially trying to find its own optimal solution [43]. Therefore, in this architecture we assume the existence of a communication channel between the eNBs to exchange the information needed to find the optimal solution. We can utilize the existing X2 interface (or Xn interface in 5G stand-alone) between the eNBs (or gNBs in 5G) to exchange information needed for RAN control.

As control is essentially the act of adapting the state of a system in response to internal and external stimuli, a controller needs levers to adjust the state of the system. Here, we exploit three primitives that are available in today's networks, which we describe next.

1) Optimize antenna tilt: There are some parameters that can be used to optimize the network coverage and capacity. Antenna tilt is one of these parameters that can be easily modified in an automatic way in order to optimize the network coverage. Antenna tilt can be defined as the inclination angle between the antenna's main beam and the horizontal plane [6]. The optimal antenna tilt value for a cell depends on the tilt values of its neighbors; too much downtilt can result in coverage holes, while too little downtilt will lead to interference with neighboring cells. We assume that only one cell can update its antenna tilt at each time step. This makes it easier to identify the impact of that change on coverage and performance.
2) Performance triggered handovers: In conventional LTE networks, handovers are mainly event-triggered. UEs measure their signal quality and report it back to the serving eNB, which uses these reports to initiate handovers when needed. Besides this, we redefine handover to include performance triggered handovers. For instance, the intelligent controller can shift users to a nearby eNB to enhance their performance. If the serving eNB is becoming overloaded, some UEs, despite good coverage, will need to switch to other eNBs for a better service and a lower delay.
3) Optimize EPS bearer rate: Evolved Packet System (EPS) bearer is defined with certain data rate according to the QoS parameters: GBR, Maximum Bit Rate (MBR), Aggregated Maximum Bit Rate (AMBR) and QCI. For GBR type applications (e.g., VoIP) which requires a constant data rate, the data rate is controlled via the GBR parameter. For non-GBR applications, such as video and best effort Internet (BE), which require a variable amount of bandwidth, the traffic is controlled by the aggregated maximum data rate (AMBR). Utilizing these data rates parameters, we can optimize the resources allocated in terms of deciding the optimum bearer rates for different users dynamically according to their QoS requirements.

V. APPROACH TO INTELLIGENT RAN CONTROL

In this section, we present the underlying network slicing scheme, the formulation of ICRAN as a deep reinforcement learning problem and its two architectures: ICRAN-C and ICRAN-D.

A. RAN Slicing Model

The allocation of radio resources to slices according to their requirements is a fundamental part of network slicing that is

usually executed at RAN. In line with the works [22], [23] discussed in Section III-A, we build our adaptive RAN slicing scheme (see Figure 2) by modifying the existing MAC scheduler and implementing a hierarchy of schedulers. Note that the LTE MAC scheduler is responsible for allocating radio resources to UEs. The frame structure of the downlink air interface contains ten subframes of 1*ms* each, which is also the transmission time interval (TTI). Besides the slice scheduler, our slicing model comprises a slicing orchestrator that determines the assignment of RBs to slices and instruct the scheduler to execute that based on the operator policy. It informs the scheduler of which slice to schedule at the current subframe based on the QCI value. If the scheduled slice has no data to transfer, the slice with the highest priority and data waiting to be transferred will be scheduled. The slice scheduler determines the slice priority based on the QoS constraints provided by the EPS bearer, which is associated with the QCI. Furthermore, the slice scheduler receives a set containing the buffer status of all UEs for each slice to determine the amount of data to be transferred. Within each slice, RBs are assigned to UEs using a conventional MAC scheduler that implements the proportional fair scheduling algorithm [44]. Our control algorithm extends this slicing scheme by adding a layer that manipulates coverage and resource allocation in response to network dynamics and performance degradation.

B. DRL Problem Formulation

We use deep reinforcement learning as a candidate solution for our problem, because DRL can accommodate the complexity of network dynamics. The future mobile networks are large-scale and complex in the sense of supporting diverse use cases which results in large state and action spaces, and the conventional control methods may not be able to find the optimal decision in reasonable time. Thus, we present the ICRAN framework based on DRL environment that comprises the real-time state of all eNBs and UEs in our slicing network. The ICRAN agent (or a set of agents) monitors the status of the eNBs including the attached UEs and network performance indicators. Based on the obtained information, the agent makes a decision whether to increase the antenna coverage, distribute the traffic loads or adjust the data rate for a specific slice to maximize the overall network performance and minimize SLA violations. The decisions are made at each control interval. The value of the control interval depends on time needed to execute the actions in the environment. The environment returns a reward, which is calculated depending on the QoS and the network efficiency, and then a new decision process will be activated. In this work, we formulate this problem in two different architectures based on the control level: 1) single agent DRL (i.e., centralized control) and 2) multi-agent DRL (i.e., decentralized control).

Formally, our centralized control problem is MDP, which is modeled as a 4-tuple $\langle S, A, r, P\rangle$, where S is the state space, A denotes the action space, r represents the reward function and P is the state transition probability for state s and action a. The state space involves the state of M eNBs and all users connected to them. The agent relies on the policy π to select actions for RAN control to maximize the reward. The task of DRL is to find the optimal policy $\pi^* : S \rightarrow P(A)$ that maximize the expected return. The return from a state s is defined as the sum of discounted future reward $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$ with a discounting factor $\gamma \in [0, 1]$. A deterministic policy returns actions to be taken in each perceived state, while the stochastic policy returns a distribution over actions. We define the Q-function, which measures the expected accumulated rewards under policy π as shown below:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_{i\geq t}, s_{i>t} \sim \mathbb{E}, a_{i>t} \sim \pi}[R_t \mid s_t, a_t] \quad (1)$$

According to the Bellman equation, the relation between the Q-function and the immediate reward can be formulated as:

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim \mathbb{E}}[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^{\pi}(s_{t+1}, a_{t+1})]] \quad (2)$$

We elaborate on the algorithm we choose to maximize this Q-function later in this sequel.

For the decentralized architecture, we extend the MDP formulation to the multi-agent setting taking into account the communication between agents [45]. We define our problem as a partially observable Markov game (POMG) for M agents comprising of a set of states S that contains the possible information of all agents, a set of actions A and a set of observations O for each agent. At each time step t, each agent takes an action $a(t)$ based on state $s(t)$, moves to new state $s(t + 1)$, receives a new observation $o(t + 1)$, and finally receives an immediate reward r. Similar to the MDP model, the agent in POMG also aims to find the optimal policy in order to maximize its expected long-term discounted reward.

Next, we elaborate on the details of our formulation.

1) State Space: The RAN controller works as an agent interacting with the network environment at every time step t. The state of the i-th eNB at time step t, $s_i(t)$, is given by:

$$s_i(t) = [DT(t), Th(t), UE_{S1}(t), UE_{S2}(t), UE_{S3}(t)] \quad (3)$$

where $DT(t)$, $Th(t)$ are the antenna downtilt and average throughput for the eNB respectively. UE_{S1}, UE_{S2}, UE_{S3} are three vectors representing the identifiers for the UEs attached to eNB from three slices S_1, S_2, S_3. For each UE in one of these three list, we use the Flow Monitor module in ns-3 to track its throughput and end-to-end delay at each time step.

2) Action Space: The action space is a vector A representing our control levers: antenna tilt optimization, traffic load balancing and traffic shaping.

- Antenna tilt optimization. For each antenna this action is defined by three discrete variables: $[a_{-}\lambda, a_0, a_{+}\lambda]$ down-tilt, no change, up-tilt the current down-tilt of magnitude λ. The minimum downtilt angle is 1° and the maximum is 14°. These are chosen to avoid excessive uptilt/downtilt [46].
- Handover: This action handovers a specific user from a specific slice from the current eNB to a nearby eNB. eNBs in LTE are interconnected with the X2 interface. If two eNBs are served by the same MME, a handover from the source to the target eNB will take place over the X2 interface. Only one UE at a time is requested to be

handed over. To maximize chances of handover success, the choice of the UE depends on the UE measurement report which contains RSRP and RSRQ values. The eNB evaluates neighboring eNBs as potential handover targets for this specific UE. The action space related to actions that optimize the traffic load for eNBs, consists of three discrete actions: $[a_{H(UE_{S1})}, a_{H(UE_{S2})}, a_{H(UE_{S3})}]$ which represent the handover of UEs from slices S_1, S_2, and S_3, respectively.

- Adjust the EPC data rate for best-effort users. To minimize the SLA violations for high priority UEs, we can reduce the aggregated maximum bit rate (AMBR) for best-effort UEs. This action is represented by $[r_{S3}]$ which is a continuous value for AMBR. We can change the rate for only one user from UE_{S3} at time *t*.

This results in a hybrid (parameterized) action space; discrete actions and continuous actions. To unify the action space, we relax the discrete actions into a continuous space using the method defined in [47]. We consider the following parameterized action space: the discrete actions are selected from a finite set $A_d = \{a_1, a_2, \ldots, a_k\}$, and each $a \in A_d$ has a set of real valued continuous parameters $X_a \subseteq \mathbb{R}$. Hence, a complete action is represented as a tuple (a, x), where $a \in A_d$ is the chosen discrete action and $x \in X_a$ is the chosen parameter to execute with action *a*. The whole action space *A* is then the union of each discrete action with all possible parameters for that action:

$$A = \bigcup_{a \in A_d} \{(a, x) | x \in X_a\} \tag{4}$$

Our DRL approach outputs a value for each of the discrete actions, concatenated with all continuous parameters, and the discrete action is chosen to be the one with the maximum output value.

Note that the previous work only considered a single category of actions, which resulted in limiting them to solving a single control problem.

3) Reward Function: The reward function is defined to guide the agent/agents to make desirable decisions in order to realize the objective of the system. Here our objective is twofold: network throughput maximization and SLA violations minimization. In response to the first objective, we include the instantaneous sum of the throughput for all *M* eNBs as the first term in Eq. (6). To achieve the second objective, we penalize the agent for any SLA violation for UEs in the system as defined below in Eq. (5) and presented as second term in the reward function.

$$p_n(t) = \begin{cases} 0, & \text{if } d_n(t) \leq D_n \wedge t_n(t) \geq T_n \\ -1, & \text{otherwise} \end{cases} \tag{5}$$

where $d_n(t), t_n(t)$ represent the end-to-end delay and throughput for UE $n \in [1..N]$ respectively. D_n, T_n are the violation threshold of latency and throughput defined for each slice in Table I. For example, if the end-to-end delay for a VoIP UE is above 100*ms* at time *t*, we consider this as a violation.

To this end, we set the reward at each time step *t* as

$$log \sum_{i=1}^{M} Th_i(t) + \sigma \sum_{n=1}^{N} p_n(t) \tag{6}$$

Algorithm 1 ICRAN-C Training Based on DDPG

Randomly initialize actor network μ and critic network Q with parameters θ^{μ} and θ^{Q} respectively.
Initialize target actor network μ_t and target critic network Q_t with parameters $\theta^{\mu t} = \theta^{\mu}$ and $\theta^{Qt} = \theta^{Q}$, respectively.
Initialize a replay buffer R with a capacity C and threshold T.
for $episode = 1, \ldots, K$ **do**
  Receive initial observation state s_0
  for $t = 1, \ldots, T$ **do**
    Select action $a_t = \mu(s_t) + \eta$ according to the current policy θ^{μ} and exploration noise η.
    Execute action a_t and observer reward r_t and new state s_{t+1}
    if $SizeofR > T$ **then**
      Sample a random minibatch of N transitions:
      $$N = R.sample(< s_i, a_i, r_i, s_{i+1} >)$$
      Set $y_i = r_i + \gamma Q^{'}(s_{i+1}, \mu^{'}(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
      Update critic Q by minimizing loss function:
      $$L = \frac{1}{N} \sum_{i} (y_i - Q(s_i, a_i|\theta^{Q}))^2$$
      Update the actor μ by applying policy gradient:
      $$\nabla_{\theta\mu} J \approx \frac{1}{N} \sum_{i} \nabla_a Q(s, a|\theta^{Q})|_{s=s_i, a=\mu(s_i)} \nabla_{\theta\mu} \mu(s|\theta^{\mu})|_{si}$$
      Update target networks:
      $$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
      $$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$
    end if
  end for
end for

We use both log function and σ, which is a positive weight, to balance between the two objectives.

C. ICRAN-C via Single-Agent DRL

In the single-agent DRL setting, owing to our continuous action space, we choose the widely used DDPG algorithm to find the optimal policy. DDPG is specifically adapted for problems with a continuous action space [48], unlike the DQN, which only works in environments with a discrete action space. DDPG is a model free algorithm because the agent cannot predict the future states of the environment without taking the action. Besides, it is an off-policy method because the policy used to improve the Q-function approximation is different from the behavior policy, used to explore the environment.

We list in Algorithm 1 the DDPG algorithm we use for training the agent. DDPG follows a critic-actor approach [49], in which an actor algorithm tries to output the best action and a critic tries to predict the value function for this action. The DDPG algorithm maintains a parameterized actor function μ to specify the current policy by deterministically mapping states

to a specific action. The critic Q is learned using the Bellman equation. First, we initialize all of the critic Q and the actor networks μ with random values of θ^{μ} and θ^{Q} respectively. Then, we start our iterative training process (Line 4). The critic network is trained to simulate the real Q-table using neural networks. The actor network is trained to generate a deterministic policy instead of the policy gradient which chooses a random action from a determined distribution. For computing optimization, the algorithm is learning in minibatches, rather than online, therefore we initialize a replay buffer *R* with fixed size in Line 3. Moreover, the replay buffer is used to store the transitions that are sampled from the environment according to the exploration policy and the tuple $<s_i, a_i, r_i, s_{i+1}>$ in Line 10. When the replay buffer is full the oldest samples are discarded. Within each time step *t*, we select an action according to the policy μ with a certain random noise (η) and we execute it as an exploration to find the best solution(Line 7). We store the transition in the replay buffer *R* (Line 10) accordingly. Note that in the forward pass, we compute a loss function (Line 12) to update the critic by minimizing such loss over the chosen random transition *i* chosen from the replay buffer *R*. We also, update the policy μ using policy gradient (Line 13), through the soft-update of the policy and critic parameters θ^{Q} and θ^{μ} respectively (Line 14).

Algorithm 2 ICRAN-D Training Based on MADDPG Algorithm for M eNBs

1: Initialize a replay buffer R with a capacity C and threshold T.
2: **for** $episode = 1, \ldots, K$ **do**
3: Initialize a random process N for action exploration
4: Receive initial observation state x
5: **for** $t = 1, \ldots$,max-episode-length **do**
6: for each eNB i Select action $a_i = \mu(s_i) + \eta$ according to the current policy θ^{μ} and exploration noise η.
7: Execute actions $a = (a_1, \ldots, a_M)$ and observe reward r and new state x'
8: Store (x, a, r, x') in replay buffer R
9: $x \leftarrow x'$
10: **for** eNB $i = 1, \ldots, M$ **do**
11: Sample a random minibatch of N transitions:
$$N = R.sample(< x^j, a^j, r^j, x'^j >)$$
12: Set $y^j = r_i^j + \gamma Q_i^{\mu'}(x'^j, a_1', \ldots, a_M')|_{a_k' = \mu_k'(s_k^j)}$
13: Update critic Q by minimizing loss function:
$$L = \frac{1}{N}\sum_j (y^j - Q_i^{\mu}(x^j, a_1^j, \ldots, a_M^j))^2$$
14: Update the actor μ by applying policy gradient:
$$\nabla_{\theta^{\mu}} J \approx \frac{1}{N}\sum_i \nabla_a Q(s, a|\theta^Q)|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^{\mu}} \mu(s|\theta^{\mu})|_{si}$$
15: **end for**
16: Update target networks for each eNB i:
$$\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$$
17: **end for**
18: **end for**

D. ICRAN-D via Multi-Agent DRL

Multi-Agent Reinforcement Learning (MARL) involves using several agents at the same time. The simplest approach in multi-agent settings is to use agents that learn and act independent of each other. We attempted this approach, by having an agent per eNB, but it did not perform well and was generally unstable. This happened because each agent's policy changes during training, resulting in a non-stationary environment. In other words, a policy change by an agent will influence the policy of the other agents and hence the lack of coordination will lead to conflicting policies. For example, handover actions produce ping pong effects and antenna tilt optimization actions produce coverage holes. Accordingly, we need to enable the agents to communicate their actions to each other. We have therefore formulated our distributed control problem as a cooperative multi-agent DRL problem, where the agents interact with each other, and their reward depends on their joint behavior. Knowing the actions taken by all agents makes the environment stationary even when policies change.

The training of multiple agents has long been a computational challenge. Since the complexity in the state and action space grows exponentially with the number of agents, even modern deep learning approaches may reach their limits [43]. If the training of agents is applied in a centralized manner, all information such as actions, observations and rewards from all gents should be sent to a centralized unit. In contrast to the centralized scheme, the training can also be handled in a distributed fashion where each agent performs local updates on and develops an individual policy without utilizing foreign information and this approach is infeasible in our work due to the non-stationarity problem. Therefore, we recognize another training scheme adopted by [50], [51]; a centralized training and a decentralized execution. This approach assumes the existence of a centralized controller that collects extra information about the agents to ease training but not used during the normal operation of the system.

To realize the multi-agent proposal, we adopt the Multi-agent DDPG framework (MADDPG) that was proposed by OpenAI in [50]. MADDPG extends DDPG into a multi-agent policy gradient algorithm where decentralized agents learn a centralized critic based on the observations and actions of all agents. Each agent has local information and local policies to train, but the centralized critic advises the agents on how to update their policies. The critic is augmented with extra information about the policies of other agents which loosen the non-stationarity of the environment. After the training is completed, the centralized critic is no longer needed; only the local actors are used in the testing phase. Here in algorithm 2, we consider a system with *M* agents, which represents eNBs, with policies parameterized by $\theta = \{\theta_1, \ldots, \theta_M\}$,

and let $\pi = \{\pi_1, \ldots, \pi_M\}$ be the set of agents' policies. $Q_i^{\pi}(x, a_1, \ldots, a_M)$ is a centralized Q-function that takes as input the actions of all agents, $a_1, \ldots, a_M$, in addition to some state information x, and outputs the Q-value for agent i. In our problem, x consists of the state space of either all eNBs or only the neighbouring eNBs (agents) to reduce the communication overhead. The first lines in the algorithms (1-7) are for parameters initialization and exploration. MADDPG uses a replay buffer to store the agent transitions (x, a, r, x') (Line 8). Then, a batch of these transitions is sampled from the experience replay to train agent i (Line 11). Line 13 is used to update an agent's centralized critic by minimizing the loss function. Note that the centralized critic uses joint information to update its parameters. Similar to DDPG in algorithm 1, MADDPG uses the deterministic policy gradient to update each of the agent's i actor parameters (Line 14). We take the gradient with respect to the actor's parameters using a centralized critic as guidance as shown below:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim p^{\mu}, a_i \sim \pi_i}[\nabla_{\theta_i} log \pi_i(a_i|s_i) Q_i^{\pi}(x, a_1, \ldots, a_M)] \tag{7}$$

The most important thing to notice is that even though the actor only has local observations and actions, the use of a centralized critic during the training phase provides information about the optimality of the agent's actions for the entire system.

E. Complexity Analysis

The complexity and implementation overhead should be kept low. This overhead may arise due to excessive signaling associated with exchange of data between ICRAN and the environment. In ICRAN-C, we assume that the network state is fully observable by the agent. During every step execution ICRAN-C collects the current state of all M eNBs in the network which results in communication overhead of $\mathcal{O}(M)$. In contrast, in ICRAN-D which is based on multi-agent DRL, we assume that the agents are communicating with each other to reach the final goal simultaneously. We propose two ways of coordination between the multiple agents; 1) exchanging information with all eNBs in the system; thus the signaling overhead for each agent is $\mathcal{O}(M-1)$. 2) exchanging information with nearby $\acute{M}$ eNBs ($\acute{M} << M$), here the signaling overhead for each agent is $\mathcal{O}(\acute{M} - 1)$. In addition to the eNB information such as antenna tilt and transmitted power, the exchanging information also contains a list of UEs' performance (i.e., throughput and delay) and their slicing profile. Hence, the size of the message between the eNB and the DRL agent in both ICRAN-C and ICRAN-D is $\mathcal{O}(P)$ where P represents the number of UEs associated with an eNB. This corresponds to a few megabytes of data for an eNB serving a million UEs.

VI. Experimental Setup

For our experiments, we used the well known ns-3 network simulator. For the reinforcement learning we used OpenAI Gym, which is a toolkit for developing RL algorithms. To integrate our network environment with Gym, we used ns3-Gym framework [52] which simplifies exchanging observations, actions and rewards between the RL agent and the network environment (See Figure 3). We implement two different architectures for ICRAN as shown in Figure 4.

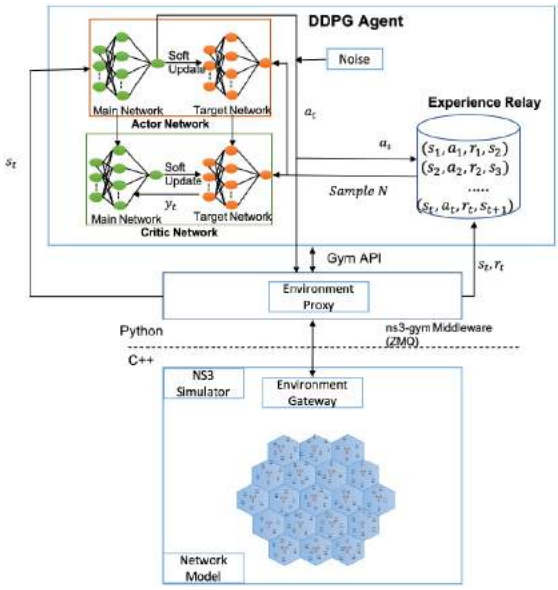


Fig. 3. Experimental Setup.

A. Network Environment

We evaluated our proposed framework in an environment consisting of 19 eNBs shown in Figure 4. We divided users evenly amongst the three slices in use: VoIP, video and best-effort. Further, we simulated VoIP using an ON/OFF model; ON is the time when users are speaking and OFF is the time when the users are not. We set the On-Time and Off-Time to 0.352 seconds and 0.650 seconds, respectively, to ensure a bit rate of 64 Kbps. The video traffic was simulated using Evalvid module[1] which streams video frames. Finally, we setup the best-effort application using the UDP echo module in ns-3. The VoIP traffic was mapped to QCI 1, the video traffic was mapped to QCI 7 and the best-effort traffic was mapped to QCI 9 and each user received traffic from only one application. Users exchange traffic with end points outside the mobile networks (i.e., the remote hosts in Figure 1).

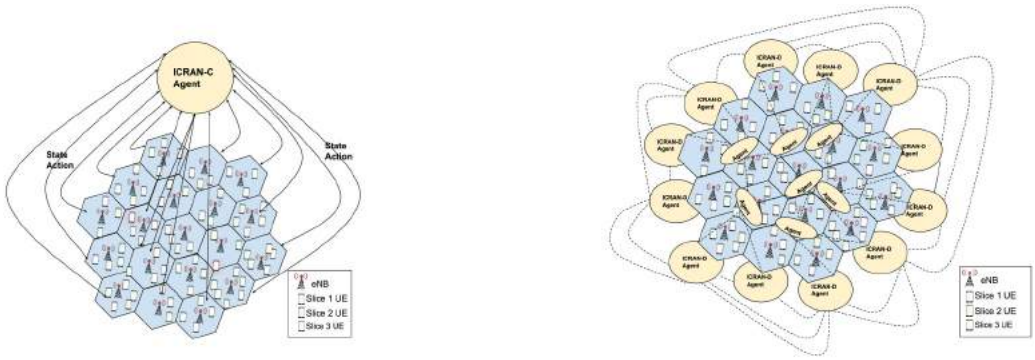
The arrival of users of each slice S followed a Poisson process with an arrival rate of $\lambda = 5$ *users/S/eNB* throughout the coverage area of the cell. 50% of the users were stationary, 25% were vehicular users moving with an average speed of 30 km/h and the rest 25% were walking at 0.8 *m/s*. Table II summarizes the simulation parameters.

B. Simulation Scenarios

For our evaluation, we considered two specific scenarios that capture extreme network conditions to help assessing the effectiveness of the proposed control mechanism.

1) Network Congestion: When the required resources by all the slices are less than the resources provided by the

[1]https://gitlab.com/gercom/evalvid-ns3



(a) ICRAN-C (b) ICRAN-D

Fig. 4. High-Level architecture for ICRAN including the network topology: a) ICRAN-C: a centralized single-agent receives a state from each eNB and chooses an action to perform to get the reward. b) ICRAN-D: partially decentralized multi agents controllers; each agent is associated with an eNB with some communication between the agents.

TABLE II
SIMULATION PARAMETERS

Parameter	Value
Number of eNBs	19
Cell Radius	250 m
eNBs Distance	500 m
Antenna type	2x2 MIMO
System Bandwidth	5MHz
eNB Tx power	30 dBm
UE Tx power	10 dBm
Mobility Model	Random Walk
Vehicular UE speed	30 km/h
Pedestrian UE speed	0.8 m/s
MAC Scheduler	Proportional Fair
Simulation Duration	500 Seconds

network, the decision making by the controller is easy. Therefore, we choose high network loads scenarios that require from the controller to obtain a complete image of the network and make intelligent decisions to achieve high end-to-end performance while minimizing SLA violations.

2) Network Failure: In this scenario, we generate some cell faults during the simulation to evaluate whether the proposed control mechanism can adapt quickly to the changes. Specifically, we randomly assign faults like excessive uptilt/downtilt to one cell. Excessive uptilt/downtilt is simulated using extreme values for the antenna downtilt which are $[0, 1]^\circ$ for excessive antenna uptilt and $[16, 15, 14]^\circ$ for excessive antenna downtilt.

C. Reinforcement Learning Setup

In this experiment, inspired by the work in [48], we used fully connected neural networks to implement the actor and critic of DDPG and MADDPG that we describe in Algorithms 1 and 2. For the actor network, we use one input layer, two hidden layers and one output layer. The neuron numbers for these layers are 16, 64, 32, 8, respectively. The critic network comprises a one input layer, one hidden layer and one output layer with neuron numbers of 24, 64, 1, respectively. For the single agent training, we set the learning rate of actor and critic in Algorithm 1 to 0.0001 and 0.001, respectively. While for multi-agent training, we use learning rate of 0.0005 and 0.001 for actor and critic networks in Algorithm 2. Additionally, for both the DDPG and MADDPG algorithms, we sample after every other 100 timesteps, and sample a batch size of 32 by episode using replay buffer size of 10000. We set τ, soft update of target, to 0.01, and the discount factor γ to 0.99 , which places more focus on the immediate reward. To balance the two components of the reward function we use $\sigma = 1.25$. In order to find suitable values for the hyper-parameters, we start with the original values proposed in [48], [50]. Then through extensive simulations and grid search, we adjusted some of the values based on the performance of the algorithms. All these parameters are summarized in Table III.

TABLE III
HYPER-PARAMETERS FOR ICRAN

Parameter	Value
Discount factor (γ)	0.99
Critic learning rate	0.001
Actor learning rate	0.0001
Replay buffer size (C)	10000
Batch size (N)	32
Soft update of target (τ)	0.01

VII. PERFORMANCE EVALUATION

A. Overview of the Evaluation

ICRAN is evaluated based on whether it achieves a network performance that satisfies the SLA requirements for slices while maximizing radio resource utilization. We conduct four stages of evaluation. The first stage looks at the training phase of ICRAN and its convergence performance (Section VII-B). The second stage investigates the decisions taken by the ICRAN agent to minimize the number of SLA violations after training and why ICRAN outperforms other decision-making approaches (Section VII-C). In the third stage, we compare ICRAN to some baselines and state-of-the-art methods in terms of physical resources utilization and SLA satisfaction (Section VII-D and Section VII-E). Finally, we study the generalized performance of ICRAN in terms of throughput and

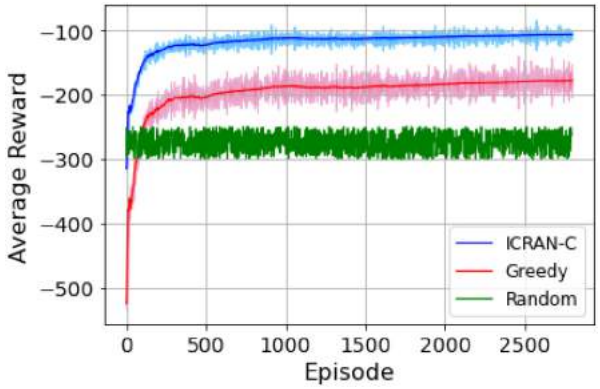


Fig. 5. The average accumulated reward that is achieved by ICRAN-C, Greedy and Random approaches. The envelope shows standard deviation.

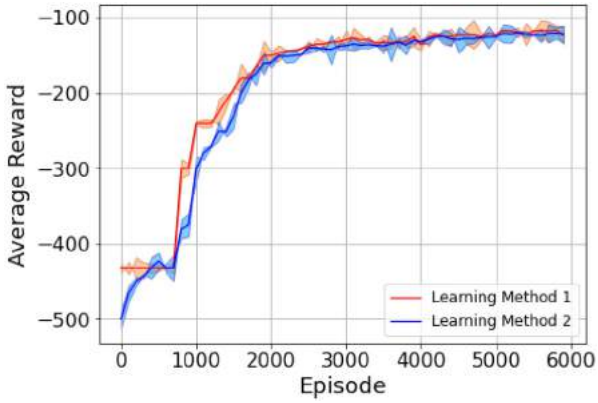


Fig. 6. The average reward of ICRAN-D for two different learning methods in the training. Method 1: Critic network of any agent receives all policy information of all agents in the system. Method 2: Critic network of any agent only receives policy information of nearby agents in the system. The envelope shows the standard deviation.

delay under different levels of network load (Section VII-E). Every scenario in this section is simulated 50 times.

B. Training Performance

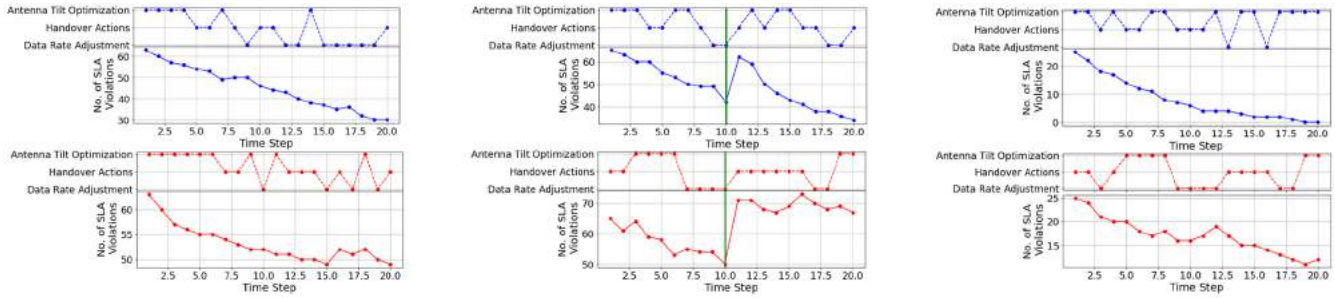
We compare the average reward and convergence performance of the ICRAN-C to those of a random and a greedy methods. The random method selects actions in the action space in a uniformly random fashion. The greedy algorithm picks the best action without taking into consideration the long-term effect of that decision. It essentially measures the immediate impact of taking an action on throughput and SLA violations. Then updates a state action table to track how effective an action is. To add some randomness, which allows the algorithm to explore the action space, we use an epsilon-greedy algorithm which adds randomness when deciding between actions. The algorithm chooses the action randomly with a probability ϵ which will make the algorithm explores other actions as well. Most of the time, the algorithm selects the best action which result in a high reward with probability $1 - \epsilon$. We try different values for ϵ and choose 0.1, which results in the best reward. Figure 5 shows that the ICRAN-C achieves a significantly higher reward than the other two alternatives. Reaching this reward level happens relatively fast, after 500 training episodes. We also note, unlike for the other two strategies, that the achieved reward exhibits low variability.

In Figure 6, we plot the average reward for the ICRAN-D. Here, we compare two learning methods. In the first method, the centralized critic uses information from all agents (red curve), while in the second learning method it uses only information from adjacent agents (blue curve). We observe a negligible difference between the two training approaches in our simulation scenario which has 19 eNB. However, we believe a larger network topology can capture the difference between the two learning methods. There are two differences between ICRAN-D and ICRAN-C. The former takes $4x$ the time the latter takes to converge. This difference holds, even if we change the learning rate. Further, the average reward of ICRAN-D is slightly lower but clearly higher than the greedy and random strategies.

C. Why Does ICRAN Outperform Other Methods?

ICRAN clearly achieves a higher reward compared to the greedy approach, which resembles the way a human operator would behave. Thus, it is important to understand the strategies that ICRAN follows to achieve this. To this end, we explore the way ICRAN and the greedy approaches behave under three different scenarios. A scenario is a single simulation snapshot, that is one realization, that lasts 20 time steps. Considering a single realization helps in gauging the impact of each action. The three scenarios are an overloaded network, a failure of an eNB in an overloaded network and a network that is loaded slightly below its full capacity. For each scenario, we track the actions that different approaches take and their impact on reducing SLA violations.

For Snapshot 1 (see Figure 7a), which represents a network congestion scenario with a network load level of 120%. ICRAN-C (blue curve) is able to reduce the SLA violations from 63 to 30, while the greedy algorithm (red curve) ends up with 49 violations. As expected, there is a mismatch between the actions taken by the two approaches. We observe that, unlike the greedy approach, some of the actions that ICRAN-C takes do not result in an immediate reduction in SLA violations. However, these actions lead to a significant drop in SLA violations in the following time steps. An example of this is the consecutive antenna tilt optimization, handover and data rate adjustment actions that are executed at time steps 7, 8 and 9. In the second snapshot in Figure 7b, we simulate the same network load level, i.e., 120% and at time step $= 10$ we introduce a failure in one eNB. The number of SLA violations jumps following the failure. The greedy algorithm handles this situation by following a sequence of handover actions to move the UEs attached to the failed eNB to nearby eNBs. This strategy minimizes the SLA violations by one at each time step. However, the ICRAN-C alternates between antenna tilt optimization and handover actions. We can observe that data rate adjustments actions are unlikely taken by ICRAN-C in this situation. ICRAN-C's strategy is to increase the coverage of the nearby eNBs' by adjusting their antenna tilt and at the



(a) Snapshot1: Network load level=120%

(b) Snapshot2: Network load level=120% with failure in one eNB at time step=10

(c) Snapshot3: Network load level=90%

Fig. 7. Action strategies taken by ICRAN-C(blue) and the greedy algorithm (red) under three testing scenarios.

same time rely on the automatic handover that the LTE system provides. This strategy helps ICRAN-C to mitigate the failure in a few time steps. Finally, the third scenario, which we depict in Figure 7c, corresponds to a network that is loaded at 90%. Accordingly, an optimal control algorithm must be able to ensure zero SLA violations. ICRAN-C indeed manages to achieve that, while the greedy algorithm fails. The main reason is that the greedy algorithm chooses to take the same action sequentially through time steps as long as the SLA violations decrease. This strategy leads to a slow decrease or increase in the number of SLA violations.

Takeaways: ICRAN outperforms the greedy approach in minimizing SLA violations. It achieves this by executing control actions that do not only focus on the immediate reduction of SLA violations. Another aspect to ICRAN is its alternation between available control actions. This strengthens the case for multi-action control strategies that prioritize long term reward. Note that the greedy strategy resembles the manual troubleshooting that is common in today's networks.

D. Resource Utilization Efficiency

Having seen that ICRAN converges to finding optimal control sequences, we now turn to evaluating ICRAN in practice. We check whether ICRAN can lead to an efficient resource utilization in comparison to two default approaches that resort to resource reservation and over provisioning. The first approach, which we call *Static Slicing* allocates 50%, 40% and 10% of the RBs to S_1, S_2 and S_3, respectively. The traditional LTE scheduler is then used to assign RBs within each slice. The second approach, *Dynamic Slicing*, besides assigns RBs to slices, it adaptively reassign RBs that are not used by a higher priority slice to a lower priority one. Note that ICRAN combines dynamic slicing and DRL (see Section V). Moreover, we compare ICRAN against two recent works from state-of-the-art. We choose these methods because they employ diverse approaches and have shown excellent performance in RAN resources management among network slices. The first work is DeepSlicing [25] that leverages a DRL algorithm, namely DDPG, to allocate the radio resources to users within a slice. For each slice there is a DRL agent that learns the optimal policy of allocating resources to users by observing the users' utility. The agent is penalized if the minimum utility requirement of users is not satisfied. Coordination of physical resources among network slices is formulated as a quadratic optimization problem aiming to maximize the sum-utility of all network slices. We implement and simulate DeepSlicing in our ns3 environment setup in Section VI; we implement three DDPG agents; each for each slice to allocate the resources to users in the network slice, i.e., action space. The reward function is penalizing the DDPG agent if the minimum utility requirement of the users is not satisfied. On the top of the DDPG agents, we solve the slices coordinating problem as a quadratic programming using the optimization tool CVXPY[2] which is an open source Python-embedded library for convex optimization problems. We train the three DDPG agents on one eNB until convergence, and then use them in all eNBs during inference. The second work is TNSM-21 [24] which is a data-driven resource management method to support RAN slicing. Based on monitoring RAN information namely, CQI reports collected from the base stations, they calculate the amount of physical resources to allocate per slice to meet the target KPIs. Similar to our proposed slicing model in Section IV, in this work the authors presented a slice orchestrator that is responsible for managing slices. Based on the UE channel quality from the CQI report, the slice orchestrator translates the CQI to the maximum data rate per RB based on [53]. Two different algorithms are being proposed for the calculation of the number of RBs per slice based on the slice requirements. For guaranteed latency slice, the number of RBs are calculated based on the queue model M/M/1/K to estimate the latency of the packets. Beside, the average packet size of the latency-constrained application and the maximum data rate provided by one RB, we can calculate the number of RB for this slice. For throughput guaranteed slice, the number of RBs is equal to (or greater than) the aggregate data rate needed by the slice for all users. We implement this algorithm in our slice orchestrator without implementing CQI overhead optimization method defined in this work. We frequently monitor the CQI value for users and map it to the maximum data rate for RB. However, we add additional case to the algorithm to cover the third best-effort slice, which assigns the remaining resource blocks to this slice.

Figure 8 shows the fraction of utilized RBs by each approach under different levels of network load. Both static and dynamic slicing achieve lower RBs utilization compared

[2]https://www.cvxpy.org/

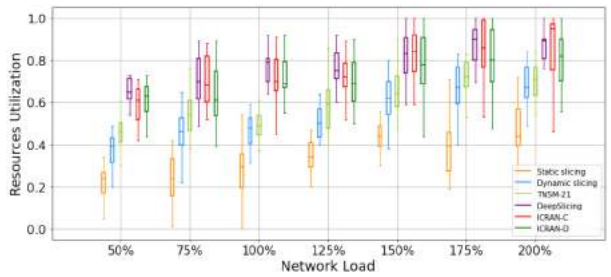


Fig. 8. Resource blocks utilization under different levels of network load.

with other methods. As expected dynamic slicing outperforms static slicing, which fails to fully utilize available resources even at lower levels of load due to its rigidity. TNSM-21 shows improvement in RBs utilization efficiency over dynamic slicing because it allocates only the required number of RBs that satisfy the slice requirements. However, because the RBs are reserved, they are not actually used until needed, thus there is room for improvement. DeepSlicing outperforms all methods at low network loads, i.e., 50%, 75% and 100% because it optimizes the RBs between the slices besides the resources allocation to individual users within the slice. The difference between DeepSlicing and ICRAN is marginal though. In a highly congested network, i.e., 200% load, ICRAN-C results in 97% median RBs utilization efficiency, which is higher than DeepSlicing's. This is because ICRAN-C attempts to perform a network-wide optimization rather than local optimization. DeepSlicing outperforms ICRAN-D, as the latter lacks the overall view of the network compared to ICRAN-C.

Takeaways: The DRL-based methods clearly outperforms the other non-DRL method. DeepSlicing and ICRAN-C demonstrate a comparable efficiency with DeepSlicing leading at lower network load (i.e., 100% or less) values and ICRAN-C leading at extreme levels of load, which is more challenging.

E. Minimization of SLA Violations

We took a closer look at the performance of ICRAN in minimizing the number of SLA violations. The two panels in Figure 9 show the number of SLA violations for the top two slices in terms of priority (*Slice1, Slice2*) achieved by different methods for different levels of network congestion. The basic slicing methods, i.e., static and dynamic demonstrate a poor performance in minimizing the number of SLA violations because they do not consider satisfaction of the slice requirements, when allocating RBs to a slice. However, we observe that DRL-based methods, namely ICRAN-C, ICRAN-D and DeepSlicing outperform TNSM-21 for both slices because this approach does not leverage any domain knowledge about the nature of the available actions, the system's state transition dynamics, and its reward function. Therefore, TNSM-21 can only allocate the minimum resources calculated by the algorithm and serve accordingly a specific number of users in *Slice1* and *Slice2*. Overall, ICRAN-C and ICRAN-D consistently outperform DeepSlicing. This difference becomes more remarkable as the network load increases. For example, when the network is 200% loaded, DeepSlicing results in 64% and 59% higher SLA violations in *Slice1* compared to ICRAN-C

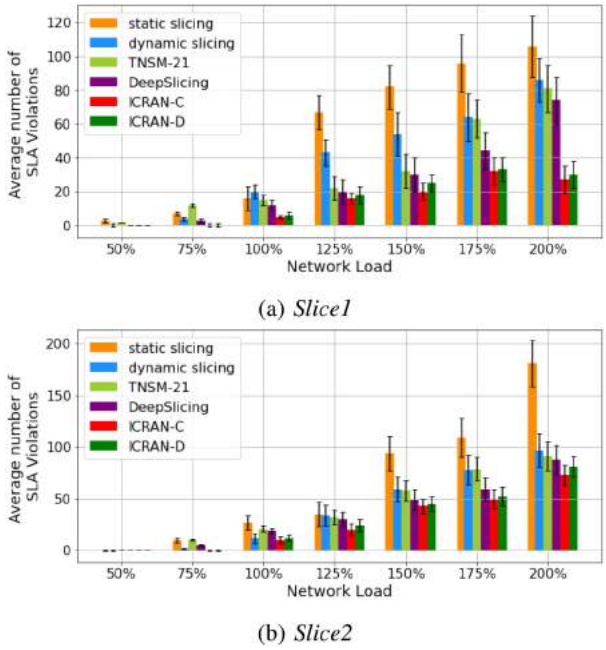


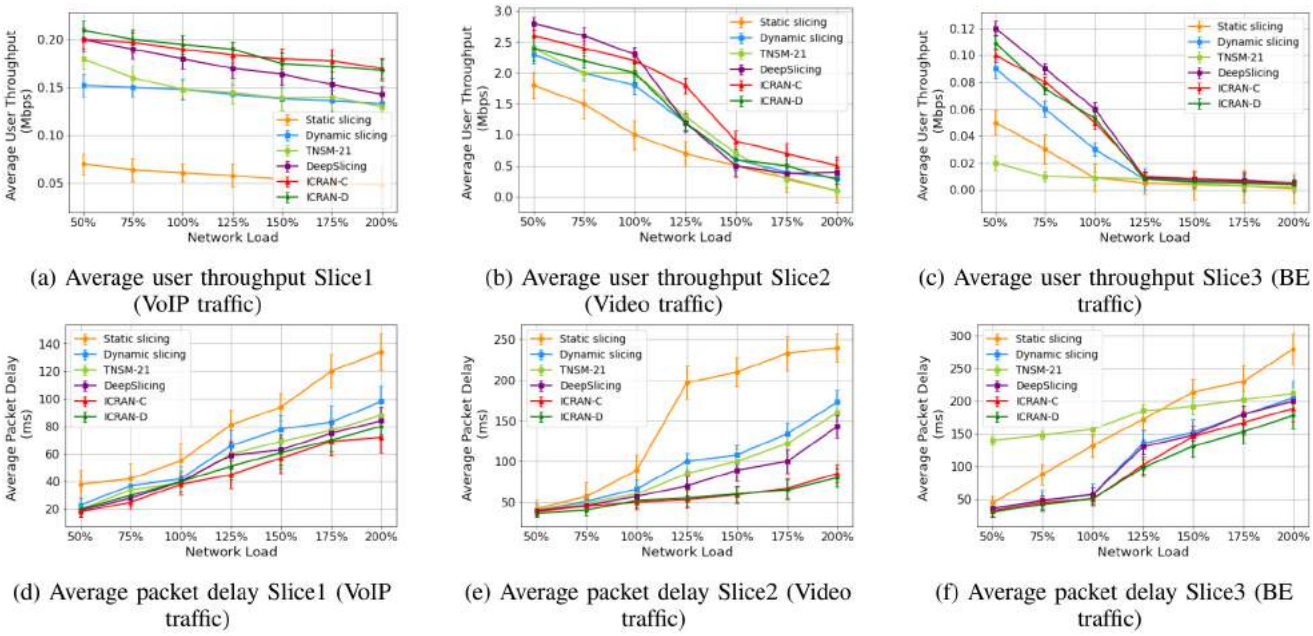
Fig. 9. Number of SLA violations, when using ICRAN-C, ICRAN-D, DeepSlicing and TNSM-21, as network load varies for (a) *Slice1* and (b) *Slice2*.

and ICRAN-D, respectively. The is because of the large action space for ICRAN. While ICRAN is giving priority to *Slice1*, DeepSlicing is treating both slices equally based on their defined SLA. However, ICRAN-C outperforms ICRAN-D in reducing the number of SLA violations, this is mainly due to the design of ICRAN-C which has a full observability of the network, while ICRAN-D has a partial observability where agents have access only to information of nearby eNBs. While the difference is generally small, both approaches respect the priority of slices. We note the trade-off between achieving efficient resource utilization and the minimization of SLA violations. ICRAN succeeds in achieving both, while DeepSlicing focuses on the former, hence its slightly better performance in RB utilization when the network is lightly to moderately loaded. However, this better performance does not translate to the best performance in terms of reducing SLA violations.

Takeaways: ICRAN outperforms the other methods in minimizing the number of SLA violations in *Slice1* and *Slice2* and considers the priority of the slices in allocating resources. The remarkable SLA violations reduction of ICRAN can be attributed to, (i) ICRAN incorporates the violation of slices' SLA into the reward function, (ii) ICRAN optimizes for the overall network which allows to utilize the physical resources efficiently to minimize the number of SLA violations for the whole network, and (iii) the larger action space that allows for quickly finding a sequence of actions that strike a balance between quality assurance and resource utilization. This confirms the need for a multi-objective reward function and a larger action space.

F. Network Performance

We now proceed to quantify the throughput and delay experienced by users in different slices, when using ICRAN, static



(a) Average user throughput Slice1 (VoIP traffic)

(b) Average user throughput Slice2 (Video traffic)

(c) Average user throughput Slice3 (BE traffic)

(d) Average packet delay Slice1 (VoIP traffic)

(e) Average packet delay Slice2 (Video traffic)

(f) Average packet delay Slice3 (BE traffic)

Fig. 10. Network performance under different levels of network load.

slicing, dynamic slicing, DeepSlicing and TNSM-21 as the network load increases. Figure 10 shows the average user throughput and average packet delay per slice for different levels of network load along with the standard deviation. All methods try to maintain a constant throughput for the VOIP slice. However, ICRAN ensures both the highest throughput of $0.143 Mbps(\pm 0.01)$ and lowest delay of $72 ms(\pm 9)$ when the network load is 200% (see Figures 10a and 10d).

In the case of video traffic, as the network load increases, the average user throughput for all methods falls (see Figure 10b). For high network loads, ICRAN-C outperforms all approaches, while DeepSlicing achieves a slightly higher throughput than ICRAN-C when the number of users per slice is low. For example, when the network load is 200%, ICRAN-C outperforms DeepSlicing by 20% in throughput. In contrast, when the network load is 50%, Deepslicing results in 7% higher throughput than ICRAN-C and 15% than ICRAN-D. This is because DeepSlicing's agent is awarded based on the maximum throughput per base station. The agent observes only the corresponding base station and cannot recognize the state of other base stations to take further actions when the network is congested. TNSM-21 exhibits the same results as dynamic slicing with little improvement. When looking at delay both ICRAN approaches clearly outperform the other four methods (see Figure 10e). When the network load is 200%, ICRAN results in an average delay that is 200%, 116%, 100% and 79% lower than that of static slicing, dynamic slicing, TNSM-21 and DeepSlicing, respectively. This is due to the reward function defined in ICRAN which directly optimizes for both throughput and delay.

Finally, for the best effort traffic, the average throughput decreases linearly, for all methods, as the network load increases from 50% to 125% (Figure 10c). ICRAN achieves a higher throughput for load levels lower than 125%. The throughput drops to a minimal level for load levels higher than 125%. For ICRAN, this is expected, since it reduces the data rate for best effort UEs to minimize the SLA violations for high priority slices. TNSM-21 achieves lower throughput regardless of the network load level, because the method is designed to allocate RBs to throughput-constrained and delay-constrained slices. The remaining RBs are allocated to best-effort UEs. There are no differences in delay between ICRAN and the other methods when the network is not loaded, we however record a slight difference, in favor of ICRAN, as the load increases (see Figure 10f). We also note that the benefit from using ICRAN is not limited to high load cases. It consistently delivers a better performance at all levels of load.

Takeaways: ICRAN demonstrates its superiority in network performance in terms of throughput and delay for the different slices. For VOIP traffic, all methods try to maintain a constant throughput, however, both ICRAN approaches achieve approximately 71%, 22%, 24% and 16% higher throughput compared with static slicing, dynamic slicing, TNNSM-21 and DeepSlicing when the network is overloaded. Also, ICRAN ensures the minimum delay in VOIP slice. In video slice, DeepSlicing shows a slight improvement in throughput over ICRAN when the number of users is low. This is due to DeepSlicing reward function that maximize for throughput at the base station. However, when the number of user increases, ICRAN performs better. For the delay, ICRAN results in the minimum delay under different network loads. ICRAN exhibits the same behavior for best-effort traffic.

VIII. Discussion

Self-driving RAN: We have demonstrated that it is possible to develop a multi-objective automated control mechanism for managing the performance in a multi-slice RAN. In the course of this work, we have identified a number of key insights that we believe are pertinent to efforts that aim to

realize a fully self-driving RAN. First, the key insight behind ICRAN is that the success in achieving the control objectives is due to the collective behaviour of all eNBs in the network. This is evident when comparing ICRAN to slicing methods and other recent works. We accordingly believe that control strategies that depend on the joint decision instead of the independent decisions of system components will be key to realizing self-driving RANs. Second, ensuring an optimal or near optimal network-wide control requires strategies that look beyond the immediate reward. Considering several time steps ahead helps avoiding actions that only result in local optimization but lead to service degradation in other parts of the network. Accordingly, heuristics and threshold-based control approaches will always likely fail in ensuring such optimality. Third, the success of ICRAN can also be attributed to the design of the reward function and the use of three different types of actions, which are both novel. Our reward function is able to guide the agent to the optimal decision through exploration and exploitation of various actions from different categories. The key takeaway point here is that a multi-objective control requires a diverse set of actions and reward functions that reflect that.

Implementation Consideration: Although, simulations confirm the effectiveness of ICRAN, it must be also practically implementable. In general, DRL does not make strong assumptions about the target system, however, we have some simulation-based assumptions, which we need to address in the real implementation. For example, we need to define the communication channels between the eNBs and the centralized agent in case of ICRAN-C and between the agents in ICRAN-D. Such channels and the respective protocols can be implemented as application level services to avoid the need for adding them to the standards. We believe that the O-RAN architecture, due to its flexibility, can ease the task of implementing such protocols [54]. Furthermore, ICRAN assumes the availability of detailed telemetry to track the state of the network. Such telemetry does not exist today but new approaches to telemetry like in-band telemetry seem promising since they balance overhead and utility [55].

Limitations and the way ahead: Even though ICRAN has succeeded in achieving its aim to optimize the overall network performance while satisfying the QoS requirements in multi-slice RAN, we highlight a number of limitations and enhancements that we plan to address in the future work. First, to support the increasing heterogeneous services and complex networks, we need to include other types of traffic with different patterns such as IoT. Second, we need to investigate the impact of the control interval on ICRAN decisions. We examine values of 1 *sec*, 5 *secs* and 10 *secs*, and decide to use 10 *secs* since this matches the time needed for performing our control actions. Shorter timings yielded poor performance results. Another issue is that our topology is relatively small, which may raise concerns about whether ICRAN can scale to much bigger networks. We believe that our preliminary results which have shown minimal differences between fully and partial observability distributed learning approaches are promising.

IX. Conclusion

In this paper, we have presented ICRAN, a novel control framework for optimizing resources utilization while minimizing SLA violations in a multi-slice RAN. Inspired by the remarkable achievements of deep reinforcement learning in solving complex control problems in highly dynamic environments such as mobile network, ICRAN comprises two DRL-based architectures: centralized ICRAN and distributed ICRAN. Through extensive simulations using ns-3, we have confirmed the substantial advantages granted by ICRAN over other slicing schemes and recent works in terms of resources utilization and QoS assurance. ICRAN is, to the best of our knowledge, the only framework that simultaneously addresses multiple RAN problems.

References

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[2] "Network Automation: Efficiency, Resilience, and the Pathway to 5G." MIT. [Online]. Available: https://www.technologyreview.com/s/613533/network-automation-efficiency-resilience-and-the-pathway-to-5g/ (Accessed: Nov. 10, 2021).

[3] A. G. Spilling, A. R. Nix, M. A. Beach, and T. J. Harrold, "Self-organisation in future mobile communications," *Electron. Commun. Eng. J.*, vol. 12, no. 3, pp. 133–147, 2000.

[4] N. Feamster and J. Rexford, "Why (and how) networks should run themselves," 2017, *arXiv:1710.11583*.

[5] L. Bonati, S. D'Oro, M. Polese, S. Basagni, and T. Melodia, "Intelligence and learning in O-RAN for data-driven NextG cellular networks," *IEEE Commun. Mag.*, vol. 59, no. 10, pp. 21–27, Oct. 2021.

[6] F. Vannella, G. Iakovidis, E. Al Hakim, E. Aumayr, and S. Feghhi, "Remote electrical tilt optimization via safe reinforcement learning," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–7.

[7] Y. S. Nasir and D. Guo, "Multi-agent deep reinforcement learning for dynamic power allocation in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2239–2250, Oct. 2019.

[8] J. Mei, X. Wang, K. Zheng, G. Boudreau, A. B. Sediq, and H. Abou-Zeid, "Intelligent radio access network slicing for service provisioning in 6G: A hierarchical deep reinforcement learning approach," *IEEE Trans. Commun.*, vol. 69, no. 9, pp. 6063–6078, Sep. 2021.

[9] Y. Kim and H. Lim, "Multi-agent reinforcement learning-based resource management for end-to-end network slicing," *IEEE Access*, vol. 9, pp. 56178–56190, 2021.

[10] A. P. Iyer, L. E. Li, and I. Stoica, "Automating diagnosis of cellular radio access network problems," in *Proc. 23rd Annu. Int. Conf. Mobile Comput. Netw.*, 2017, pp. 79–87.

[11] G. Brockman *et al.*, "OpenAI gym," 2016, *arXiv:1606.01540*.

[12] "ns-3 Network Simulator." [Online]. Available: https://www.nsnam.org/ (Accessed: Nov. 10, 2021).

[13] M. Iwamura, "NGMN view on 5G architecture," in *Proc. IEEE 81st Veh. Technol. Conf. (VTC Spring)*, 2015, pp. 1–5.

[14] M. Richart, J. Baliosian, J. Serrat, and J.-L. Gorricho, "Resource slicing in virtual wireless networks: A survey," *IEEE Trans. Netw. Service Manag.*, vol. 13, no. 3, pp. 462–476, Sep. 2016.

[15] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 2429–2453, 3rd Quart., 2018.

[16] S. Wijethilaka and M. Liyanage, "Survey on network slicing for Internet of Things realization in 5G networks," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 957–994, 2nd Quart., 2021.

[17] Y. Wu, H.-N. Dai, H. Wang, Z. Xiong, and S. Guo, "A survey of intelligent network slicing management for industrial IoT: Integrated approaches for smart transportation, smart energy, and smart factory," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 2, pp. 1175–1211, 2nd Quart., 2022.

[18] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.

[19] ETSI NFV, "Network functions virtualization: An introduction, benefits, enablers, challenges & call for action," Darmstadt, Germany, SDN OpenFlow World Congr., White Paper, 2012.
[20] T. Taleb, A. Ksentini, and B. Sericola, "On service resilience in cloud-native 5G mobile systems," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 3, pp. 483–496, Mar. 2016.
[21] "O-RAN: Towards an open and smart RAN," Alfter, Germany, O-RAN Alliance, White Paper, Oct. 2018.
[22] D. Nojima *et al.*, "Resource isolation in RAN part while utilizing ordinary scheduling algorithm for network slicing," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, 2018, pp. 1–5.
[23] R. Shrivastava, K. Samdanis, and A. Bakry, "On policy based RAN slicing for emerging 5G TDD networks," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2018, pp. 1–6.
[24] S. Bakri, P. A. Frangoudis, A. Ksentini, and M. Bouaziz, "Data-driven RAN slicing mechanisms for 5G and beyond," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4654–4668, Dec. 2021.
[25] Q. Liu, T. Han, N. Zhang, and Y. Wang, "DeepSlicing: Deep reinforcement learning assisted resource allocation for network slicing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, 2020, pp. 1–6.
[26] A. Abouaomar, A. Taik, A. Filali, and S. Cherkaoui, "Federated learning for RAN slicing in beyond 5G networks," 2022, *arXiv:2206.11328*.
[27] N. C. Luong *et al.*, "Applications of deep reinforcement learning in communications and networking: A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, 4th Quart., 2019.
[28] A. Feriani and E. Hossain, "Single and multi-agent deep reinforcement learning for AI-enabled wireless networks: A tutorial," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1226–1252, 2nd Quart., 2021.
[29] L. Lei, Y. Tan, K. Zheng, S. Liu, K. Zhang, and X. Shen, "Deep reinforcement learning for autonomous Internet of Things: Model, applications and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1722–1760, 3rd Quart., 2020.
[30] C. She *et al.*, "A tutorial on ultrareliable and low-latency communications in 6G: Integrating domain knowledge into deep learning," *Proc. IEEE*, vol. 109, no. 3, pp. 204–246, Mar. 2021.
[31] F. Tang, Y. Kawamoto, N. Kato, and J. Liu, "Future intelligent and secure vehicular network toward 6G: Machine-learning approaches," *Proc. IEEE*, vol. 108, no. 2, pp. 292–307, Feb. 2020.
[32] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep reinforcement learning for mobile edge caching: Review, new features, and open issues," *IEEE Netw.*, vol. 32, no. 6, pp. 50–57, Nov./Dec. 2018.
[33] A. M. Seid, G. O. Boateng, B. Mareri, G. Sun, and W. Jiang, "Multi-agent DRL for task offloading and resource allocation in multi-UAV enabled IoT edge network," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 4, pp. 4531–4547, Dec. 2021.
[34] J. Li, H. Gao, T. Lv, and Y. Lu, "Deep reinforcement learning based computation offloading and resource allocation for MEC," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2018, pp. 1–6.
[35] Y. Ren, A. Guo, C. Song, and Y. Xing, "Dynamic resource allocation scheme and deep deterministic policy gradient-based mobile edge computing slices system," *IEEE Access*, vol. 9, pp. 86062–86073, 2021.
[36] A. M. Seid, G. O. Boateng, S. Anokye, T. Kwantwi, G. Sun, and G. Liu, "Collaborative computation offloading and resource allocation in multi-UAV-assisted IoT networks: A deep reinforcement learning approach," *IEEE Internet Things J.*, vol. 8, no. 15, pp. 12203–12218, Aug. 2021.
[37] J. Pérez-Romero, O. Sallent, R. Ferrús, and R. Agustí, "Knowledge-based 5G radio access network planning and optimization," in *Proc. Int. Symp. Wireless Commun. Syst. (ISWCS)*, 2016, pp. 359–365.
[38] "5G-Lena Module." [Online]. Available: https://5g-lena.cttc.es (Accessed Nov. 15, 2021).
[39] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, and J. Wang, "Large-scale measurement and characterization of cellular machine-to-machine traffic," *IEEE/ACM Trans. Netw.*, vol. 21, no. 6, pp. 1960–1973, Dec. 2013.
[40] A. Aghmadi, I. Bouksim, A. Kobbane, and T. Taleb, "A MTC traffic generation and QCI priority-first scheduling algorithm over LTE," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, 2015, pp. 1–6.
[41] P. H. A. Rezende and E. R. M. Madeira, "An adaptive network slicing for LTE radio access networks," in *Proc. Wireless Days (WD)*, 2018, pp. 68–73.
[42] H.-S. Chuang, S.-L. Hsieh, and C.-F. Wu, "A channel-aware downlink scheduling scheme for real-time services in long-term evolution systems," in *Engineering Innovation and Design*. Boca Raton, FL, USA: CRC Press, 2019, pp. 337–343.
[43] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: A survey," *Artif. Intell. Rev.*, vol. 55, pp. 895–943, Apr. 2021.
[44] F. Capozzi, G. Piro, L. A. Grieco, G. Boggia, and P. Camarda, "Downlink packet scheduling in LTE cellular networks: Key design issues and a survey," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 678–700, 2nd Quart., 2012.
[45] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings*. Amsterdam, The Netherlands: Elsevier, 1994, pp. 157–163.
[46] A. Gómez-Andrades, P. Muñoz, E. J. Khatib, I. de-la Bandera, I. Serrano, and R. Barco, "Methodology for the design and evaluation of self-healing LTE networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 8, pp. 6468–6486, Aug. 2016.
[47] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half field offense: An environment for multiagent learning and ad hoc teamwork," in *Proc. AAMAS Adaptive Learn. Agents (ALA) Workshop*, 2016, pp. 1–7.
[48] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning," 2015, *arXiv:1509.02971*.
[49] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2000, pp. 1057–1063.
[50] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," 2017, *arXiv:1706.02275*.
[51] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," 2016, *arXiv:1605.06676*.
[52] P. Gawłowicz and A. Zubow, "ns-3 meets OpenAI gym: The playground for machine learning in networking research," in *Proc. 22nd Int. ACM Conf. Model. Anal. Simul. Wireless Mobile Syst.*, 2019, pp. 113–120.
[53] *LET; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical Layer Procedures, V.15.2.0, Release 15*, 3GPP standard TS 36.213, Oct. 2018.
[54] A. Garcia-Saavedra and X. Costa-Pérez, "O-RAN: Disrupting the virtualized RAN ecosystem," *IEEE Commun. Stand. Mag.*, vol. 5, no. 4, pp. 96–103, Dec. 2021.
[55] L. Tan *et al.*, "In-band network telemetry: A survey," *Comput. Netw.*, vol. 186, Feb. 2021, Art. no. 107763.

Azza H. Ahmed (Member, IEEE) received the master's degree from the University of Nottingham in 2012. She is currently pursuing the Ph.D. degree with the Simula Metropolitan Center for Digital Engineering, Oslo, Norway. Her research interests include communication networks management and control, network performance optimization, network automation, and machine learning to solve networks problems.

Ahmed Elmokashfi received the Ph.D. degree from the University of Oslo in 2011. He is a Research Professor with the Simula Metropolitan Center for Digital Engineering in Norway. He is currently working as the Head of the Center for Resilient Networks and Applications, which is part of the Simula Metropolitan Center, which is funded by the Norwegian Ministry of Transport and Communication. In particular, he focused on studying resilience, scalability, and evolution of the Internet infrastructure; the measurement and quantification of robustness in mobile broadband networks; and the understanding of dynamical complex systems. Over the past few years, he has been leading and contributing to the development, operation and management of the NorNet testbed infrastructure, which is a countrywide measurement setup for monitoring the performance of mobile broadband networks in Norway. His research interests lie in network measurements and performance.

Article IV

Evang, J. M., Ahmed, A. H., Elmokashfi, A. and Bryhni, H. (2022, 26. July). **Crosslayer Network Outage Classification Using Machine Learning.**
Applied Networking Research Workshop 2022 (ANRW'22).
DOI: **https://doi.org/10.1145/3547115.3547193**

Crosslayer Network Outage Classification Using Machine Learning

Jan Marius Evang
marius@simula.no
SimulaMet, OsloMet
Oslo, Norway

Azza H. Ahmed
azza@simula.no
SimulaMet, OsloMet
Oslo, Norway

Ahmed Elmokashfi
ahmed@simula.no
SimulaMet
Oslo, Norway

Haakon Bryhni
haakonbryhni@simula.no
SimulaMet
Oslo, Norway

ABSTRACT

Network failures are common, difficult to troubleshoot, and small operators with limited resources need better tools for troubleshooting. In this paper, we analyse two years of outages from a small global network for high-quality services. Then, we develop a machine learning model for outage classification that can be set up with little effort and low risk. We use passive Bidirectional Forwarding Detection (BFD) data to classify Layer2 problems and add active packet loss data to classify other problems. The Layer2 problems were classified with a 99% accuracy and the other problems with 40%–100% accuracy. This is a significant improvement when we observe that only 35% of the customer cases we studied received any Reason for Outage (RFO) response from the Customer Support Centre.

CCS CONCEPTS

• **Networks** → **Public Internet**; **Network measurement**; • **Computing methodologies** → **Supervised learning**.

ACM Reference Format:
Jan Marius Evang, Azza H. Ahmed, Ahmed Elmokashfi, and Haakon Bryhni. 2022. Crosslayer Network Outage Classification Using Machine Learning. In *Applied Networking Research Workshop (ANRW '22), July 25–29, 2022, PHILADELPHIA, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3547115.3547193

1 INTRODUCTION

Today's Internet comprises a group of small, medium, large and extra large networks as far as geographic presence and traffic volume are concerned. The end-to-end network service is produced following a three-layer model that is similar to the lower levels of the OSI reference model [1].

ANRW '22, July 25–29, 2022, PHILADELPHIA, PA, USA

ACM ISBN 978-1-4503-9444-4/22/07…$15.00
https://doi.org/10.1145/3547115.3547193

Table 1: Manually classified causes.

Class	Cause	Count
MultiLoss	Multiple Layer2 providers	870
CogentLoss	Cogent's network	556
Customer	Customer's equipment	411
TeliaLoss	Telia's network	316
Layer3	Layer3 only	222
InternMaint	Internal maintenance	114
Optic	3dB Optical change	74
ProvMaint	Provider maintenance	70
EquinixLoss	Equinix Cloud Fabric	58
SubseaCable	Subsea cable outages	42
EquipFail	Equipment failure	40
FiberCut	Fiber cut in provider network	39
Layer1	Leased Layer1 lines	18
Metro	Metropolitan area links	18
DoS	Denial of Service attacks	4

A few large providers sell Layer2 capacity based on the global mesh of Layer1 optical fibres, which are used by Layer3 providers to compose end to end services.

This layered architecture is exposed to various types of faults, such as physical fiber faults, equipment faults, planned maintenance and malicious attacks. Our data shows that the Layer1/Layer2 service has a high number of faults (see Table 1). Smaller networks that lease Layer1/Layer2 services need to quickly attribute such faults and report them to the respective providers. This is important for two reasons. First, it can help shorten the resolution time. Second, faults must be reported during the incident to be acknowledged according to the Service Level Agreements (SLAs).

Unlike large networks with sizable organization and abundant resources, small and medium network operators have a much smaller Network Operations Centre (NOC) with limited resources and staff. A typical small-medium NOC either operates a single enterprise network or is a speciality Internet Service Provider (ISP) providing a service to select customers in a narrow business area or in a geographic area.

Smaller NOCs often have a small but highly demanding customer base, for instance their co-workers in an enterprise, people in their

own geographic area or specialized service providers. This makes detecting and isolating faults very important yet a demanding task.

The NOC usually has automatic network monitoring systems in operation, but they can suffer from large numbers of both false positives (alerts without a real fault) and false negatives (faults that do not generate an alarm). This often causes true positives to be overlooked [2]. In an outage event where one component in the network has failed, causing interruption to network traffic, an overwhelming amount of log messages and alerts will be arriving from different monitoring systems. This makes the NOC waste time and effort to find the real cause. In other cases, a problem may not be noticed until customers complain. Customer Support (CS), may not have enough information to respond to a customer case because the NOC is busy troubleshooting. Alarm Consolidation systems exist but they suffer from high complexity [3], narrow field [4] or high compute requirements [5]. In this work, we tackle these problems by developing a generic model to assist NOCs and CSes. We leverage supervised learning to assist in classifying different outages. For classification, we use the Support-Vector Machine model (SVM) [6]. Our system is two-stage. In the first, it discriminates Layer1/Layer2 problems from Layer3 ones. Here, we identify a set of easily to collect metrics that can help achieving this in an efficient manner. In the second, it classifies Layer3 problems based on their root causes.

The research in [7] claims that supervised learning for fault classification is often suffering from low quality of training data, but in our research we have access to precise outage data, including root cause data.

Our system requires minimal changes to the network, and has a minimal impact on networking equipment and computing power. We also demonstrate that our proposed system is implementable and can be used to assist an existing provider efficiently.

With the system developed here, the NOC will speed up troubleshooting, quickly create trouble tickets with the providers, and the CS will improve customer satisfaction by giving informed feedback to all customer support cases. Compared to similar systems such as [5], investments in time and equipment are small, changes to configuration is minimal, and causes are successfully predicted with an f1-score of 0.99 for Layer2 cases and f1-score of 0.66 for other cases (see Section 4.1). Without the tool, only 35% of the cases received any outage report from CS.

2 RELATED WORK

Various works have used machine learning and other statistical methods for attributing faults for specific network protocols, however, there is still lack of work that leverages logs from different layers, and predict causes across network layers.

Existing research such as [3] implements a complex system of user defined scenarios, while they do not require detailed knowledge of the underlying system, they cannot detect problems outside manually defined failure scenarios. Our labeled data and two-stage approach makes classification of known faults across all layers possible and efficient, and the feedback loop handles new fault classes. Moreover, several projects [4, 8–12] examine how very detailed measurements of optical signal strength can be used to gain knowledge about the underlying Layer1 links. However, these methods require measurement of q-factor [13] telemetry [14] which is unavailable to higher layers providers.

The research in [15] also uses customer tickets for anomaly detection, but focuses only on Layer1 and last mile. The authors in [16] analyse Layers 2-7, while we analyse backbone Layers 1-3, and a common "No issue" class for any problems in other layers or outside the backbone network. Unlike [17], our system does not need any knowledge about the underlying network, only the manual feedback needs this.

Some commercial service providers have implemented systems for anomaly detection in system logs, for instance [18]. These systems have the advantage that they analyse the existing logs, and therefore are easy to start using, however, there is a high risk of exposing confidential information to a third party. In our system, only the feedback loop will have any confidentiality risk.

Finally, the authors in [5] analyse traffic by using a distributed Apache Storm [19] system in combination with data obtained from the Netflow [20] protocol. This puts extra stress on the networking equipment [21] and demands much more storage and CPU power, making it undesirable unless Netflow is already used for other purposes. BFD, on the other hand, is usually implemented in hardware.

The objective in this paper is to fill the gap and use simple data logs from various layers together with customer support data to classify outages in a fast and easily-implementable low-impact solution.

3 METHODOLOGY

3.1 Description of system

Our system consists of a data collection unit, a classification model (See Section 3.4), and alert and feedback units as shown in Figure 1.

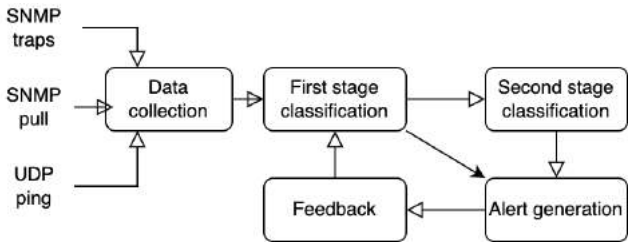


Figure 1: Our proposed outage classification system.

We collected the measurements from a global network covering 12 cities around the world, which we depict in Figure 2. The network uses three different Layer2 service providers to interconnect its points of presence (PoPs). The first is Telia VPLS, which is full-mesh Layer2 switched network based on VPLS/ELAN [22] over their global backbone network. The VPLS service supports Q-in-Q switching [23], so individual point-to-point VLANs [24] are configured, with each VLAN having member ports from only two cities. The second is Cogent L2C, which is a point-to-point MPLS [25] based service where multiple point-to-point Layer2 links are provided over the same physical interface. The third is Equinix Cloud Exchange Fabric (ECXF), which is a service of multiple point-to-point Layer2 links over the same physical interface.

The network is set up with IS-IS + BFD [26, 27] as Interior Gateway Protocol (IGP). The IGP makes sure that in case of issues on one link or device, customer traffic is automatically re-routed to an alternative path.

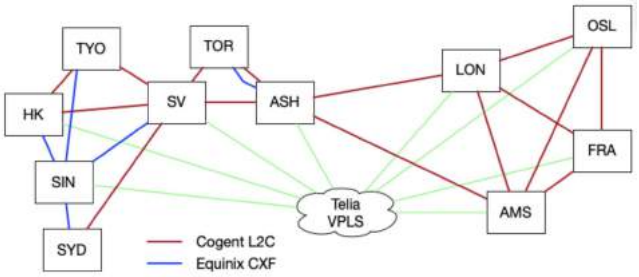


Figure 2: Network design.

3.2 Data description

In this work, we collected data from the monitored ISP over two years (2019-08-08 to 2021-10-01). Below, we list these measurements alongside their description.

Optical signal strength measurements. Every 30 minutes, optical received signal strength for the local link to the provider was read via SNMP [27]. There were 12 total outages on 5 interfaces, 88 drops in optical strength of more than 3dB on 14 interfaces and 53 increases of more than 3dB on 10 interfaces.

Interface error counters. Every 10 minutes, interface error counters for all devices were logged by SNMP polling. No interface errors were recorded for Layer2 switch ports, because any such errors would have been revealed and corrected during pre-production testing.

Buffer overflows/Tail-drops. Every 10 minutes, the buffer overflow/tail-drop counters were logged by SNMP polling. Only two interfaces showed tail drops, altogether 40 incidents. The NOC had especially amended this risk by over-provisioned the network to handle network traffic peaks without packet loss.

Layer2 packet loss data. In each of the 11 cities shown in Figure 2, 2 probe Virtual Machines (VMs) were set up. Our probe software is based on OpenNetNorad [28], which we rewrote in C to improve performance and reduce CPU consumption. This "pinger" transmits 100 UDP 64 byte packets every 0.5 seconds and waits for responses, and the "ponger" immediately returns any received packets to the sending IP address. The number of lost packets is then recorded. Packets are transmitted from 100 different UDP ports to detect any issues related to link aggregation or Equal Cost Multipath (ECMP) within the provider network. In addition to the probes measuring point-to-point (P2P) loss over the Layer2 links, a full mesh of probes (FM) were set up to measure the Layer3 service. One or more lost UDP packets in a 0.5*sec* interval generates one loss report. There were 196 million loss reports for 36 different pairs of probe VMs. These were pre-processed to 717352 unique events (see Section 3.3).

BFD traps. For each point-to-point link or VLAN, BFD (Bidirectional Forwarding Detection) [29] is configured to send one packet every 100*ms*. If 3 packets in a row are lost, the link is declared down and an SNMP trap message is sent to a collector. SNMP trap data is passively collected and stored in a database for later processing. The IS-IS protocol also receives BFD events and takes care of re-routing traffic.

Software crash logs. There were 62 instances of software crash/core-dumps incidents on the routers and switches. Most of these did not cause any interruption to network traffic since the Forwarding Engines were still operational.

Configuration change logs. Configuration change logs indicate which piece of equipment was configured and when. Also a textual description of the work was performed.

Customer complaints data. The customers' systems have strict network requirements for latency, packet loss and jitter. Customer cases were raised upon any violation of these requirements. The data was anonymized and made available for this work. During the period, there were 19399 customer cases, of which 8120 were related to the network. The complaints were reduced to 2855 unique cases on 21 different paths.

Customer service response data. For each customer case, CS analysed logs and provided a Reason For Outage (RFO) if possible. Out of 2855 cases, 1014 (35%) received RFO from CS, 109 of these were "no issue found".

Manual analysis of customer reports. We looked at all available data for each customer reported case and determined the reason for the incident. In most cases the cause was in a Layer2 provider's network. For other cases the cause could be determined more precisely from CS responses. The results are presented in Table 1.

In some cases, there were losses in multiple providers at the same time, which may be caused by either an (undetected) failure in the monitored network, a larger failure that impacted multiple providers, short traffic peaks that caused packet loss and therefore triggered a re-routing to another provider and subsequent loss there, or could be just a coincidence.

Multiple customer complaints received within a 5-minute interval were counted as one case. Still, a single root cause could cause multiple cases over a longer time. Some incidents were caused by planned or unplanned maintenance. These were recorded as cases, if they caused customer complaints even when the customer had been informed ahead of time.

The 713857 events that did not correspond to customer cases were not manually analysed.

3.3 Data preprocessing

The data used for the Machine Learning algorithm was BFD SNMP events (BFD), point-to-point UDP pings (P2P) and full-mesh UDP pings (FM). The other data was used only in the manual classification process of all the cases. The result of the manual classification was used to train the supervised machine learning system.

Due to small delays in detection and collection of test data, the resolution of the timestamps had to be reduced to match events from different sources. Each measuring point was added as a separate feature, with an aggregation of the number of such events per minute. One minute aggregation was chosen as a trade-off between fast detection and data size. For the BFD and P2P data, the measuring points were each link, for the FM data, the measuring points were

the unique pairs of PoPs. This resulted in a dataset of 717352 unique events and 2855 unique cases. The features were 47 BFD, 32 P2P and 125 FM. The classes with < 4 cases were omitted.

3.4 Model description

We tested both Multilayer Perception neural networks (MLP) and SVM. SVM had both shortest processing time and highest classification accuracy, and is used in this paper. The data was split 75:25 into a training dataset and a testing dataset, and we tuned the hyperparameters using grid search. The optimal kernel was the Radial Basis Function (RBF) kernel with $C = 150$ and $\gamma = 7.5 \times 10^{-5}$. SVM is in general resistant to overfitting and we verified this by ShuffleSplit [30] and saw that the f1-score remained the same.

The first stage classification used only BFD data for classifying the largest and most precisely defined classes, i.e. the Layer2 provider cases. The output was five classes. One per each Layer2 provider, A fourth class that involve cases where more than one Layer2 provider, and one "Layer3" class for cases which were not caused by Layer2 events. A large number of events were processed in the first stage, but since fewer features were used, processing requirements were greatly reduced. The second stage classification used BFD, P2P and FM data for the Layer3 class to give an indication of the root cause. Since a much smaller subset of events was processed in this stage, the addition of more features did not lead to a large increase in processing power requirement. See also Section 4.4.

The feedback loop is used by NOC/CS when a prediction has failed, to manually correct the case label in the data and re-train the model.

After training the two machine learning models on the case data, the trained models were applied to all events, to see what knowledge could be gained.

4 PERFORMANCE EVALUATION

4.1 Evaluation metrics

We used the precision, recall and f1-score to assess our classifier.

For each class, the precision is the number of correctly predicted cases divided by the total predictions in that class. Recall is the number of correctly predicted cases divided by the number of true cases in that class. F1-score is the harmonic mean of precision and recall [31].

To visually evaluate the output of the classification process, we plot the Confusion Matrices. These show how well the model was able to assign a correct "predicted label" to each class of "true labels". The diagonals of the matrices show the correct predictions.

4.2 Accuracy and Feature importance

We performed the first classification stage initially by including all features, which resulted in a precision of 0.89, a recall of 0.89 and an f1-score of 0.92 (see the confusion matrix is in Figure 3a).

Using only BFD features showed much better scores for Layer2 cases, but did (as expected) not distinguish between Layer3 and Customer issues as seen in the confusion matrix in Figure 3b and scores in Table 2. Total f1-score was now 0.99 with a combined Customer+Layer3 class . Further, repeating the first stage while including only FM and only P2P gave poor results with f1-score 0.35 for FM and and f1-score of 0.26 for P2P (see Figures 3c and 3d).

Table 2: First stage evaluation, based on BFD.

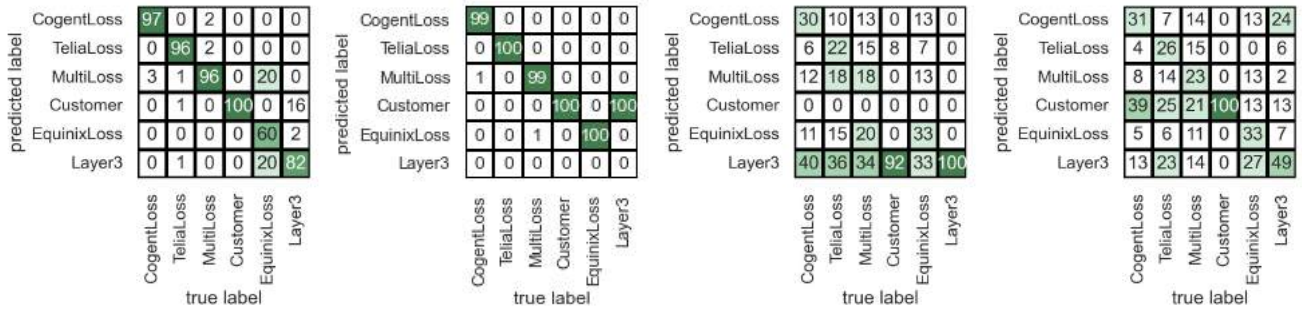
class	precision	recall	f1-score
CogentLoss	1.00	0.99	0.99
TeliaLoss	1.00	1.00	1.00
MultiLoss	0.99	0.99	0.99
EquinixLoss	0.88	1.00	0.94
Customer	0.65	1.00	0.79
Layer3	0.00	0.00	0.00

Table 3: Second stage prediction scores (Based on BFD+P2P+FM)

class	precision	recall	f1-score
InternMaint	0.65	0.72	0.68
Optic	0.75	0.43	0.55
ProvMaint	0.33	0.55	0.41
SubseaCable	0.92	0.92	0.92
EquipFail	0.40	0.40	0.40
FiberCut	0.75	0.64	0.69
Layer1	0.83	1.00	0.91
Metro	1.00	0.43	0.60
DoS	1.00	1.00	1.00

The BFD analysis contained only 4 misclassifications: 2 MultiLoss events classified as EquinixLoss were caused by two unrelated coinciding loss events where the EquinixLoss event affected multiple Equinix links, and 2 CogentLoss events classified as MultiLoss were multiple coinciding Cogent events. The analysis including all features added the capability of distinguishing between Layer3 loss and Customer loss, at the expense of requiring more computing time and adding more "noise" to the various Layer2-classifications. Still, we see a relatively small number of misclassifications (14 misclassified and 429 correctly classified Layer2 events).

For the second stage, the events that were identified by the first stage classification were removed, and a new supervised classification was attempted for the remaining events. After hyperparameter tuning, this classification showed an f1-score of 0.66. The size of the dataset in this analysis is only 437 cases with 204 features, and the results were not as good as for the first stage, but a reasonable suggestion for a root cause might still provide valuable input to the NOC's troubleshooting process. Figure 4 and Table 3 show the confusion matrix and classification score for stage 2, respectively. We can clearly see that determining the exact root cause can be hard for a few types of failures. For instance, ProvMaint and InternalMaint events may cause a wide variety of different error symptoms, that may be indistinguishable from the other classes. Interestingly, subsea cable cuts (f1-score 0.92) and fiber cuts (f1-score 0.69) had relatively good classification scores, even though these were thought to be difficult to distinguish. A point for future study might be to understand why.



(a) All features (b) BFD features only (c) FM features only (d) P2P features only

Figure 3: First stage classifications

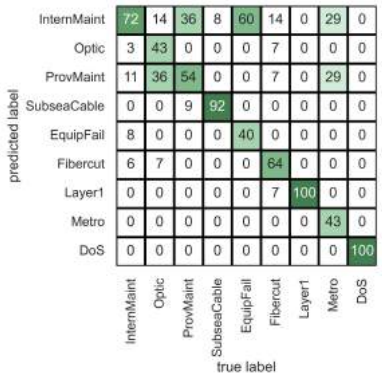


Figure 4: Confusion matrix for second stage classification

Table 4: First stage data of the Layer2 cases and predictions

class	support cases	extrapolated cases
CogentLoss	556 (19.4%)	49997 (7.0%)
TeliaLoss	316 (11.1%)	64859 (9.1%)
MultiLoss	870 (30.5%)	47227 (6.6%)
EquinixLoss	58 (2.0%)	15346 (2.1%)
Customer+Layer3	633 (22.2%)	536428 (75.1%)
Other	14.8%	

4.3 Extrapolation

Using the first stage model, BFD-trained on the cases with well known cause and symptoms, we ran a prediction on all the events where we did not get any customer complaints, to get an idea of how common the various types of problems are in these events. The very high f1-score of the model fitted on the complaint-data means that the predictions on the non-complaint-data will be highly relevant for our research. However, selection bias in that some hidden class of outages never leads to complaints might reduce the accuracy of the extrapolation.

For the first stage model, the results can be seen in Table 4. The most interesting observation is that the "Customer+Layer3" classification is much more common than in the cases where the customers filed complaints. (75.1% of the events, versus 22.2% of the cases). This means that the test network does a good job of hiding Layer3 problems from customers, and Layer2 problems are more likely to cause customer complaints, but still only 0.4% of all events caused customer cases.

Table 5: Second stage data for the Layer3 cases and predictions

class	cases	predictions
InternMaint	114 (27.1%)	185169 (34.5%)
Optic	74 (17.6%)	216591 (40.4%)
ProvMaint	70 (16.7%)	26883 (5.0%)
SubseaCable	42 (10.0%)	19522 (3.6%)
EquipFail	40 (9.5%)	18308 (3.4%)
FiberCut	39 (9.3%)	58099 (10.8%)
Layer1	18 (4.3%)	10450 (1.9%)
Metro	19 (4.5%)	1329 (0.2%)
DoS	4 (1.0%)	77(0.01%)

The "MultiLoss" class is only 6.6% of the events in the non-complaint dataset, vs 30.5% of the complaint-cases. This indicates that the network is better at hiding Layer2 problems in a single provider, and problems affecting multiple providers are more likely to generate customer complaints.

Further, we used the model fitted on the Second stage data from the cases, and made a prediction using only the "Customer+Layer3" class from the first stage non-complaint events. Applying the second stage model to the non-complaint data gives an indication that Internal maintenance and Optic events are less likely to cause customer complaints than the other classes, but the size of the dataset and the lower accuracy of the model makes these results much less certain. See Table 5.

4.4 Processing performance

BFD is implemented in hardware on our routers and do not put any load on the routers' CPU. To compare, Netflow would cause 15%-20% CPU impact according to [5], which matches our own experience. SNMP traps produced by the routers using the lowest priority processes, and all data is transmitted blindly using UDP, also reducing processing. Our ML processing on an M1 Pro 10 core

CPU took <1sec. The amount of stored data for the ML system is low. For each BFD trap we store timestamp+link-id and for each UDP measurement we store timestamp, source/destination address and loss percentage.

5 DISCUSSION

Our analysis of two years of outage data shows that a two-stage classification system is well suited to classify network outages, providing the NOC with useful predictions on where to start troubleshooting, and providing CS with RFO for all cases with a much better success rate than the observed 35% of CS responses during the period of the study. BFD data exhibits their high importance in the classification. In contrast, although the active P2P and FM raw data provides very precise measurements, they are not highly contributing to discriminating features in the classification model.

One important shortcoming is that we do not have latency measurements. But as our analysis reveals, BFD SNMP traps are very good indicator of problem types and location, so latency changes would probably not have a great impact on this result. Moreover, in this work, customer complaints are the only source for determining whether a packet loss event is regarded as an outage. Only the cases that are received as customer complaints are analysed in detail. This means that some outages may be overlooked if the customer did not complain, and some complaints may be groundless (i.e caused by other factors than the test network). A customer complaint is only counted as a network outage if the timestamp is reported as within 60 seconds of an internal packet loss or BFD trap event.

There are many features that show some correlation, which might disturb the machine learning classification model since one event is likely to affect multiple features. But since the features have a large geographic spread, and since there are many features, a certain degree of correlation should not cause problems for our analysis.

Model Drift (MD) is another consideration. During the 2 years of data collection, there were continuous changes to both the network topology, the routing protocols and the customer's monitoring system. MD may have degraded our analysis, in that patterns for the various classes of events change over time. However, this will also reflect more accurately a real-life situation. The results prove that our first stage analysis was not significantly affected by MD. In the future, we plan to gain insight into how our model may degrade, for instance by temporal cross validation, and how to rectify it through a system for retraining while running in the production. A future improvement, especially for the second stage, would be to also report the second ranked classification for an outage.

Another very important practical consideration is the difference in complexity of gathering the data for the first stage and the second stage. The passive BFD data used for the first stage is very easy to collect. Most networks already use the BFD protocol as a part of the IGP protocol, but very few actually gather the SNMP trap data from BFD. The changes and risk to the network will be small, the BFD events are already being detected, so the only change is to generate SNMP trap messages and set up one central location to store these (optionally a second location for redundancy.)

The active P2P and FM raw data are very accurate and provides very precise measurements, but showed less precision in case classification, and the system used to gather this data is much more expensive in management and computing power.

6 CONCLUSION

We have developed a system that Network Operations Centres and Support Centres for smaller operators can use in a failure situation. Using minimal resources, we passively collect BFD data and classify the Layer2 events to an f1-score of 0.99. By adding a second stage with active monitoring to collect UDP ping data we predict other types of root cases with a 0.66 f1-score. Our analysis interestingly shows that BFD features, which are the easiest to collect, give the best results for outage classification.

REFERENCES

[1] J. Day and H. Zimmermann, "The osi reference model," *Proceedings of the IEEE*, vol. 71, no. 12, pp. 1334–1340, 1983.

[2] B. AlAhmadi, L. Axon, and I. Martinovic, "99% false positives: A qualitative study of soc analysts' perspectives on security alarms," *USENIX Association*, 2021. [Online]. Available: https://ora.ox.ac.uk/objects/uuid:0be05f6b-7470-4210-acb6-2018d5dc6ca0

[3] K. Appleby, G. S. Goldszmidt, and M. Steinder, "Yemanja—a layered fault localization system for multi-domain computing utilities," *Journal of Network and Systems Management*, vol. 10, pp. 171–194, 2004.

[4] T. Christopoulos, O. Tsilipakos, G. Sinatkas, and E. E. Kriezis, "On the calculation of the quality factor in contemporary photonic resonant structures," *Opt. Express*, vol. 27, no. 10, pp. 14 505–14 522, May 2019. [Online]. Available: http://opg.optica.org/oe/abstract.cfm?URI=oe-27-10-14505

[5] Y. Du, J. Liu, F. Liu, and L. Chen, "A real-time anomalies detection system based on streaming technology," in *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics*, vol. 2, 2014, pp. 275–279.

[6] R. Soentpiet *et al.*, *Advances in kernel methods: support vector learning*. MIT press, 1999.

[7] S. Ayoubi, N. Limam, M. A. Salahuddin, N. Shahriar, R. Boutaba, F. Estrada-Solano, and O. M. Caicedo, "Machine learning for cognitive network management," *IEEE Communications Magazine*, vol. 56, no. 1, pp. 158–165, 2018.

[8] C. Natalino, A. di Giglio, M. Schiano, and M. Furdek, "Root cause analysis for autonomous optical networks: A physical layer security use case," in *2020 European Conference on Optical Communications (ECOC)*, 2020, pp. 1–4.

[9] L. Shu, Z. Yu, Z. Wan, J. Zhang, S. Hu, and K. Xu, "Low-complexity dual-stage soft failure detection by exploiting digital spectrum information," in *45th European Conference on Optical Communication (ECOC 2019)*, 2019, pp. 1–4.

[10] C. Delezoide, P. Ramantanis, L. Gifre, F. Boitier, and P. Layec, "Field trial of failure localization in a backbone optical network," in *2021 European Conference on Optical Communication (ECOC)*, 2021, pp. 1–4.

[11] Ujjwal, J. Thangaraj, and A. A. Dias Barreto, "Accurate qot estimation for the optimized design of optical transport network based on advanced deep learning model," *Optical Fiber Technology*, vol. 70, p. 102895, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1068520022000785

[12] C. Miao, M. Chen, A. Gupta, Z. Meng, L. Ye, J. Xiao, J. Chen, Z. He, X. Luo, J. Wang, and H. Yu, "Detecting ephemeral optical events with OpTel," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 339–353. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/miao

[13] "Recommendation O.201: Q-factor test equipment to estimate the transmission performance of optical channels," International Organization for Standardization, Geneva, CH, Standard, 2003.

[14] [Online]. Available: https://www.juniper.net/documentation/us/en/software/junos/interfaces-telemetry/index.html

[15] J. Hu, Z. Zhou, and X. Yang, "Characterizing Physical-Layer transmission errors in cable broadband networks," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*. Renton, WA: USENIX Association, Apr. 2022, pp. 845–859. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/hu

[16] J. Iurman, F. Brockners, and B. Donnet, "Towards cross-layer telemetry," in *Proceedings of the Applied Networking Research Workshop*, ser. ANRW '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 15–21. [Online]. Available: https://doi.org/10.1145/3472305.3472313

[17] R. R. Kompella, J. Yates, A. Greenberg, and A. C. Snoeren, "IP fault localization via risk modeling," in *2nd Symposium on Networked Systems*

Design & Implementation (NSDI 05). Boston, MA: USENIX Association, May 2005. [Online]. Available: https://www.usenix.org/conference/nsdi-05/ip-fault-localization-risk-modeling
[18] [Online]. Available: zerbium.com
[19] The Apache Software Foundation, "Apache storm," https://storm.apache.org/.
[20] E. B. Claise, "Cisco systems netflow services export version 9," Internet Requests for Comments, RFC Editor, RFC 3954, 8 2004, http://www.rfc-editor.org/rfc/rfc3954.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3954.txt
[21] "Netflow services," p. 74, 2003.
[22] Metro Ethernet Forum, "Ethernet services definitions - phase 2," 4 2008.
[23] "Provider bridges, ieee std. 802.1ad," 2005.
[24] "Bridges and bridged networks, ieee std. 802.1q-2018," 2016.
[25] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," Internet Requests for Comments, RFC Editor, RFC 3031, 1 2001, http://www.rfc-editor.org/rfc/rfc3031.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3031.txt
[26] International Organization for Standardization, *Information technology — Telecommunications and information exchange between systems — Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the connectionless-mode network service*, ISO/IEC10589:2002 ed. Vernier, Geneva, Switzerland: International Organization for Standardization, 2015. [Online]. Available: https://www.iso.org/standard/30932.html
[27] D. Katz and D. Ward, "Bidirectional forwarding detection (bfd) for ipv4 and ipv6 (single hop)," Internet Requests for Comments, RFC Editor, RFC 5881, 6 2010.
[28] Facebook Inc, "Opennetnorad," https://github.com/fbsamples/OpenNetNorad, 2017.
[29] R. Presuhn, "Version 2 of the protocol operations for the simple network management protocol (snmp)," Internet Requests for Comments, RFC Editor, STD 62, 12 2002, http://www.rfc-editor.org/rfc/rfc3416.txt. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3416.txt
[30] Q.-S. Xu and Y.-Z. Liang, "Monte carlo cross validation," *Chemometrics and Intelligent Laboratory Systems*, vol. 56, no. 1, pp. 1–11, 2001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169743900001222
[31] N. Chinchor, "MUC-4 Evaluation Metrics," in *Proceedings of the 4th Conference on Message Understanding*, ser. MUC4 '92. USA: Association for Computational Linguistics, 1992, p. 22–29. [Online]. Available: https://doi.org/10.3115/1072064.1072067

Article V

Fida*, M., Ahmed*, A. H., Dreibholz, T., Ocampo, A. F., Michelinakis, F. I and Elmokashfi, A. **Bottleneck identification in cloudified mobile networks based on distributed telemetry.** Submitted to Conference on emerging Networking EXperiments and Technologies (CoNEXT 2022).
URL: **https://www.techrxiv.org/articles/preprint/Bottleneck_Identification_in_Cloudified_Mobile_Networks_based_on_Distributed_Telemetry/22100546**

Bottleneck identification in cloudified mobile networks based on distributed telemetry

MahRukh Fida*
mrukh@glos.ac.uk
University of Gloucestershire
Cheltenham, UK

Azza H. Ahmed*
azza@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

Thomas Dreibholz
dreibh@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

Andrés F. Ocampo
andres@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

Foivos I. Michelinakis
foivos@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

Ahmed Elmokashfi
ahmed@simula.no
Simula Metropolitan Center for Digital Engineering
Oslo, Norway

ABSTRACT

Cloudified mobile networks, such as 5G, are expected to deliver a multitude of services to several slices in parallel, while having reduced capital and operating expenses. Therefore, 5G mobile systems, need to ensure that the Service Level Agreements (SLA) of customized end-to-end sliced services are met. This requires monitoring the resource usage and characteristics of data flows at the virtualized network components and interfaces of its cloud mobile network, as well as tracking the performance of its radio interfaces and user equipments (UEs). A central monitoring architecture is impossible to support millions of UEs though. This paper, proposes a distributed telemetry framework in which UEs act as early warning sensors. Upon flagging an anomaly, the cloudified mobile network activates its machine learning model to attribute the cause of the anomaly. For root cause analysis we employ active, passive and in-band telemetry in our network. Our framework achieves an impressive performance of 85% F1 score in detecting anomalies caused by different bottlenecks, and an overall 89% F1 score in attributing these bottlenecks.

1 INTRODUCTION

As mobile networks are transforming into multi-service infrastructures, quality assurance is increasingly both central and important. In the near future, these networks are expected to carry, beside today's best effort traffic, a multitude of use cases with stringent requirements e.g. IoT, industrial automation and highly interactive multiverse traffic [10]. Ensuring that the different tenants are accommodated goes beyond over-provisioning base stations and fibre links to quickly detecting and remediating performance degradations. Note that all these steps need to apply to both, a very granular scope (i.e. a few users covered by a single base station) as well as geographical scopes in an increasing coarseness. Detecting performance degradation necessitates the timely collection of representative telemetry, the automation of flagging any performance degradation and root cause attribution, and finally the design of an effective control system that reconfigures the affected network elements.

This paper proposes an approach for timely collecting telemetry, detecting and attributing issues in mobile networks. Unlike data centre networks, architectures for collecting telemetry in mobile networks have received little attention. Differences between the two types of networks, especially the challenging radio interface, make adopting data centre approaches a long shot at best. We tackle this by leveraging user equipment (UEs) as early warning sensors. A UE performs a simple anomaly detection and informs the network in case an anomaly is detected. On the network side, we experiment with a diverse set of metrics that includes active, passive and in-band telemetry. Once a problem is flagged by a set of UEs, a network controller will use a supervised machine learning algorithm to identify the root cause of the anomaly. However, a classical shortcoming of such approaches is the fact that training may not be comprehensive enough to include all types of anomalies. To address this, we devise a novel method to single out new types of anomalies and use them to further improve the system.

We have built a software defined virtualised testbed that resembles a cloudified mobile network and used it to assess our telemetry architecture. To this end, we have introduced a set of performance bottlenecks both in the radio access network (RAN) and the core. Our system achieved an impressive performance of 85% F1 score in detecting bottlenecks by the end-user and overall 89% F1 score in attributing the bottlenecks based on network measurement.

* These authors contributed equally to this work.

This paper makes three key contributions:

(1) First, it proposes a new approach for anomaly detection in mobile network that can leverage millions of end devices attached to its edge i.e. UEs, along with a complete architecture.
(2) Second, it proposes a viable solution to the problem of using supervised ML for classifying problems with potentially yet unlearned classes in the context of mobile networks.
(3) Finally, we have empirically evaluated our approach in the lab.

2 BACKGROUND AND MOTIVATION

5G network is expected to be flexible and programmable, with the ability to support instantiation of services across heterogeneous network components and virtualised infrastructures. It allows service providers to deploy and operate different services as separate network slices in parallel, with the guarantee of slice isolation and predefined quality level, namely a service level agreement (SLA).

To meet SLAs, it is critical for the service providers and network operators to constantly monitor various components of the network. It is required to identify faults in the network and resolve the issues in a timely manner, so as to minimize service downtime. In addition to infrastructure performance, it is essential to monitor applications from an end-user's perspective. Monitoring of end-user application performance informs if customer expectations are met as well as to single out any radio issue, that may go undetected by monitoring at rest of the mobile network.

Our goal in the study is to design a framework that can both flag events causing performance degradation to an end service, and to attribute and localize it with minimal system overhead. We term the events causing end-to-end performance degradation as bottlenecks.

Literature study shows that machine learning (ML) can be leveraged to automate detection of the bottlenecks at run time [31]. For both the detection and attribution of the bottlenecks, a supervised learning method is employed as in [9], so to arrive at the root cause of performance degradation in a mobile network. These methods are dependent upon labeled training samples, and fail when an unlearned issue arises.

Secondly, the accuracy of a supervised ML model depends on the availability of data. To trigger and attribute issues that deteriorate performance quality of an end service, it is essential to provide a holistic view of the network system to the ML model. It includes monitoring of both the network infrastructure and the applications from an end-user's perspective. Monitoring performance of end services, can be done with crowdsourcing [14]. It includes passive as well as active monitoring of the quality of service (QoS) features e.g. network coverage and quality of experience (QoE) features such as page load time in web-browsing applications. Crowdsourcing is expensive in terms of data caps, especially when a crowdsourced end-user has to periodically report the monitored features, to a central monitoring system.

As far as the network infrastructure is concerned, passive measurement strategy best suits it when its components are operated/controlled by a single entity. It involves recording and analyzing the actual user traffic to understand network usage trends. In situations where it is not possible to select capture points freely, or some links or connecting devices cannot be monitored passively, active probing is used. This method injects test traffic into the network to find faults or issues within the network. Active probes are controllable in terms of when and what network features to measure. It, however, burdens the probed devices and links with additional data. Passive measurements do not inject additional data, but monitoring all the traffic flows can be expensive in terms of memory and processing resources of the monitoring devices. Secondly network administrators and operators typically utilise a client–server monitoring framework, in which case even the passive measurements incur communication overhead other than management and privacy issues. It is usually impractical for all the training data to be uploaded to the central controller with an increasing network congestion. As an alternate to the traditional passive monitoring strategies, one can apply the In-band Network Telemetry (INT) [19] method that is used in datacenters. INT is passive monitoring system implemented with Programming Protocol-independent Packet Processors (P4) [3]. Being programmable, it allows a centralised network controller not only to configure the measurement frequency and to change the monitored features on the fly, but also to adjust the monitoring granularity to per user-end, per-link, per-flow down to a packet-level.

In the light of the above discussed potentials and challenges, we aim for a distributed monitoring architecture that may leverage local (or semi-local) learning with minimal monitoring overhead.

3 SYSTEM INFRASTRUCTURE

The hardware and software infrastructure of our mobile network testbed is illustrated in Figure 1.

3.1 Mobile Network

Main part of the network is an Enhanced Packet Core (EPC), containing the four basic components [7]:

(1) Home Subscriber Server (HSS), for managing the network subscriber accounts;

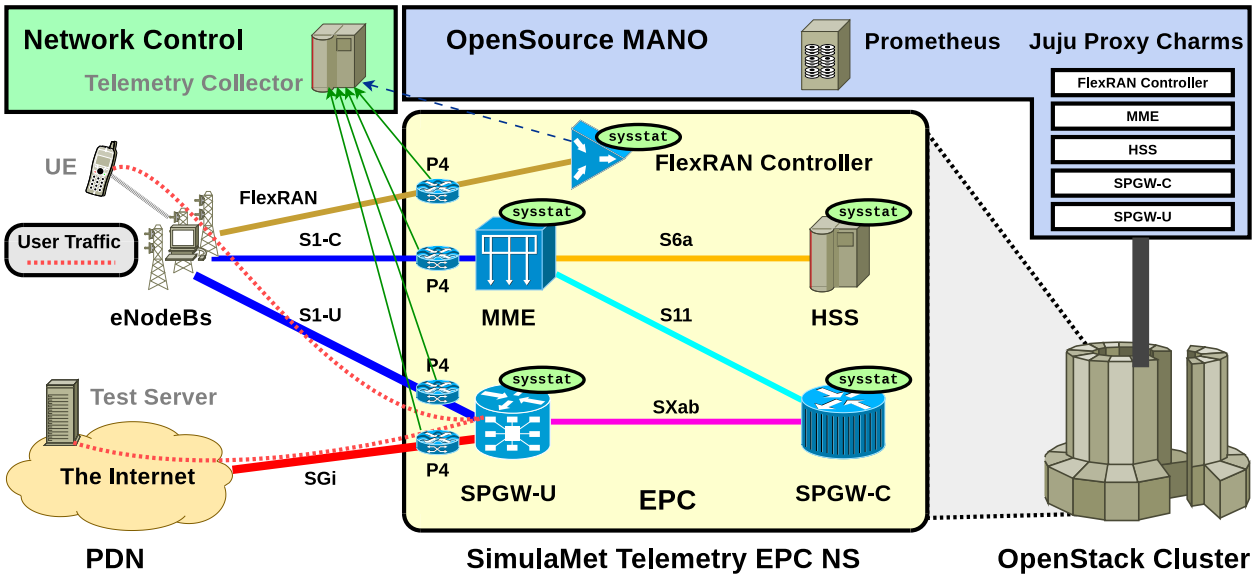


Figure 1: The Infrastructure of the Network System.

(2) Mobility Management Entity (MME) for managing the attachment of Evolved Node Bs (eNodeB, i.e. base station) and User Equipments (UE, e.g. smartphones or modems);
(3) Control Plane of the Packet Data Network Gateway (SPGW-C), for managing access to a Public Data Network (PDN, i.e. the Internet);
(4) User Plane of the Packet Data Network Gateway (SPGW-U), for forwarding user traffic between UEs and the PDN.

HSS, MME, SPGW-C and SPGW-U are using the open source implementation from OpenAirInterface* (OAI) [6, 7]. In addition to four EPC components, we also deploy FlexRAN†, particularly a FlexRAN Controller to manage the eNodeB parameters and provide fine-granular metrics from the eNodeBs.

The eNodeB is deployed using the open source implementation from OpenAirInterface [24]. In addition, the software defined radio (SDR) Ettus USRP B210 provides both the antennas and the Radio Unit (RU), which converts radio waves into digital waveforms.

Clearly, managing the components of a complex setup manually is not straightforward. Therefore, we deploy Open Source MANO‡ (OSM) as orchestration platform for Network Function Virtualisation (NFV). Basically, OSM performs [29, Chapter 1]:

- Composition of VNFs into Network Services (NS);
- Instatiation of NSs and their VNFs in an underlying Network Function Virtualisation Infrastructure (NFVI) as so-called Virtual Deployment Units (VDU), which are virtual machines and/or containers;
- Run-time configuration (e.g. initial installation, run-time change of parameters, reconfiguration) of the VDUs;
- Monitoring of the VDUs (details in Subsection 3.2);
- Scaling (i.e. increasing/decreasing the number of instances) and removal of VDUs.

Currently, our setup is running OSM "Release EIGHT" on Ubuntu 18.04 "Bionic Beaver". OSM uses Juju§ for managing the VDUs. That is, for each VDU, Juju maintains a separate container that controls the VDU. Each container runs the Juju Charm of the corresponding component, which is a custom Python program to implement the component-specific control functionalities. In our setup, we use OpenStack [25] "Stein" on Ubuntu 19.04 "Disco Dingo" as NFVI to host the VDUs, which are instantiated as virtual machines in two OpenStack compute nodes. The VDUs run Ubuntu 18.04 "Bionic Beaver", while all other VDUs run Ubuntu 20.04 "Focal Fossa".

3.2 Telemetry Components

Clearly, as part of the components orchestration, OSM already provides two ways of monitoring the deployed NSs:

*OpenAirInterface: https://www.openairinterface.org.
†FlexRAN: https://mosaic5g.io/flexran.
‡Open Source MANO: https://osm.etsi.org.
§Juju: https://jaas.ai.

(1) By using features of the NFVI (i.e. by Ceilometer and Gnocchi in OpenStack [25]);
(2) By Juju Charms, that run customised monitoring code as part of the configuration service managing the VDUs.

However, this monitoring only covers coarse metrics [36, 38] – like CPU utilisation, per-interface packet and byte counters, etc. – and does not represent the quality of services features of user data traffic. Particularly, there is no information about user flows (e.g. TCP connections, etc.) of users. Packet and byte counters only represent the aggregation of *all* users and their flows. We aim at a vendor-independent, "standardised" solution for passive monitoring of the per-flow user data traffic with P4[¶] [5] based software switches. P4 provides a standardised language for programming packet processors, i.e. switches, which can be compiled for different target devices. Currently, we deploy P4 software switches, using the Behavioral Model Version 2 (BMv2) Simple Switch software implementation[‖]. However, once available, it would be straightforward to just replace them by more powerful, off-the-shelf P4 hardware switches. In our testbed, as shown in Figure 1, we have P4 switches for the four important interfaces (actually: internal networks):

(1) S1-C, between eNodeBs and MME (network control traffic);
(2) S1-U, between eNodeBs and SPGW-U (encapsulated user traffic);
(3) SGi, between SPGW-U and PDN (decapsulated user traffic);
(4) FlexRAN, between eNodeBs and FlexRAN Controller (only FlexRAN control traffic).

Particularly, the user traffic is handled on the S1-U interface, where it is tunnelled via GPRS Tunnelling Protocol (GTP), and on the SGi interface, where it is "normal" traffic without encapsulation. It should also be noted that SGi traffic uses the public IP address of the SPGW-U. The SPGW-U performs Network/Port Address Translation (NAT/PAT) between internal addresses used by the UEs, and the public SPGW-U address. Inside the tunnel, traffic therefore uses the internal addresses of the UEs.

We programmed the P4 switches to attach, process and remove custom telemetry data to/from packets running over them. For example, the S1-U switch can attach information to a user packet (in the GTP tunnel), and forward the modified version of the packet to the Telemetry Collector server. Then, the SGi switch can do the same, and remove the telemetry information before forwarding the packet into the Internet.

[¶]P4: https://p4.org.

[‖]Behavioral Model Version 2 (BMv2) Simple Switch: https://github.com/nsg-ethz/p4-learning/wiki/BMv2-Simple-Switch.

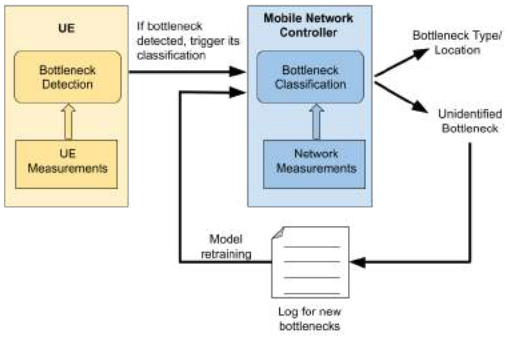


Figure 2: Proposed bottleneck identification system.

The Telemetry Collector can correlate the two packet snippets (on the flow identifiers that come with the INT fields), and generate performance metrics. Note that the actual outgoing packet into the Internet does *not* contain telemetry data any more, i.e. the privacy of the user is not getting compromised.

With the INT information of the packets, collected by the P4 switches at S1-U and SGi, the Telemetry Collector can track both the characteristics of user data traffic and the status of the P4 switches such as congestion at a port. Depending on the processing power on P4 switches and the Telemetry Collector Server, the system may be configured to only handle a subset of the packets or flows, for creating samples (e.g. only every n-th packet or flow or only flows of certain representative users, etc.).

4 BOTTLENECK IDENTIFICATION ARCHITECTURE

Our goal is to use the monitoring at UE and telemetry components described in Subsection 3.2 for bottleneck identification. Figure 2 depicts the distributed architecture of our proposed bottleneck identification and classification system. It comprises of three stages:

(1) System monitoring
(2) Bottleneck detection
(3) Bottleneck classification

An elaboration of each stage is offered in the following.

4.1 System Monitoring

Based on the design considerations discussed above, we monitor communication system both at a user side (i.e. at UE) and at the rest of the mobile network.

Monitoring at UE records QoE-based features of the applications running on the end-device. Being dependent on an application, this can be page load time and throughput

Type	Description	Examples of features
Monitoring at UE		
Active monitoring from UE	Ping to Server	Average RTT, Packet loss percentage
Passive monitoring of an application service	Example: Downlink TCP iPerf to Server	Transfer (MB), Bitrate (Mbps), Jitter(ms)
Passive monitoring of network coverage	Example: NetMonitor App	RSRP, RSRQ, RSSNR
Monitoring of network links		
Active monitoring of S1-U and SGi links	Ping from eNodeB to SPGW-U	Average RTT, Packet loss percentage
	Ping from SPGW-U to the test server	Average RTT, Packet loss percentage
Passive monitoring by P4 switches at S1-U and SGi	INT on user data traffic and status of the switches	Packet count of a flow, *packet_length*, *deq_qdepth*, *deq_timedelta*, hitter
Monitoring of network resources		
Passive monitoring resource usage at SPGW-U and FlexRAN controller	Using *sar* utility of *Sysstat* to collect CPU and memory and disk input output activit	CPU usage features (i.e. %user, %system, %iowait, %steal, %idle), Memory usage features (i.e. kbmemfree, kbavail, kbmemused, permemused, kbbuffers, kbcached, kbcommit, percommit, kbactive, kbinact, kbdirty) and I/O features (such as tps, rtps, wtps, dtps) [11]
Monitoring by FlexRAN controller		
Control commands from northbound RESTful API of FlexRAN controller	To obtains configurations and statistics of eNB(s) and their UE(s)	We record features specified by `flex_cell_config`, `flex_ue_config` and `flex_ue_stats_report`**.

Table 1: Monitoring the mobile network.

for a web-browsing service, while for a streaming video it can be delay, jitter and throughput. For our test scenario, we take downlink TCP iPerf from the test server as a user data session, the quality features of which, shown in Table 1, are monitored passively. The downlink data transfer is performed at the maximum bandwidth of the end-to-end path. UE tracks RTT to the test server by sending UDP ping messages, every second. This active measurement is done to monitor delay in the mobile network. Lastly UE passively monitors the radio coverage quality via NetMonitor[††], every second. NetMonitor collects Reference Signal Received Power (RSRP), Reference Signal Received Quality (RSRQ) and Received Signal to Noise Ratio (RSSNR).

To monitor system resources, performance, and use of the network components such as SPGW-U and FlexRAN controller we run Sysstat utility[‡‡] [11] periodically, every 5 seconds. The FlexRAN controller provides a northbound RESTful API for issuing control commands and for obtaining statistics and reports for the connected base stations using simple HTTP requests[§§]. We run *curl -X GET http:// 127.0.0.1:9999/ stats/ manager/ all* command, every 5 seconds. This gets the RAN configuration and status for the current TTI for all eNBs connected to this controller. It reports on the configuration of eNB(s) and UE(s), and statistics about Medium Access Control (MAC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP) layers.

To monitor user-data traffic, network links and status of switching devices, we utilize the P4 switches of S1-U and SGi interfaces (see also Figure 1). The switches create clones of the passing by packets and adds an additional header of "IP options" [28, Subsection 3.1] with telemetry fields that may both comprise of status of the P4 switches such as their buffer occupancy and flow/packet characteristics such as arrival time of a packet. The metrics of INT can be programmed depending upon the monitoring requirement. These cloned (mirrored) packets are then sent to the Telemetry Collector (shown in Figure 1), for further analysis. For

[††] NetMonitor is an Android app from https://vavsoftware.ru.

[‡‡] Sysstat: https://bencane.com/2012/07/08/sar-sysstat-linux-performance-statistics-with-ease.

[§§] FlexRAN northbound API: https://mosaic5g.io/apidocs/flexran/#api-Stats-GetStatsHumanReadable.

our test scenario we compute four parameters for INT on each of the two switches. These are (1) *packet count of a data flow*, (2) *hitter* which is a Boolean metric that assesses if the packet is part of a bursty traffic. We have taken switch queue size of 5000 or more bytes as an indicator of a bursty traffic, (3) *deq_timedelta* that measures in microseconds the amount of time a packet spends in P4 switch queue, and (4) *deq_qdepth* which indicates the depth of the switch queue when the packet was dequeued, in units of number of packets. The last two parameters are derived from the struct *standard_metadata* of the $P4_{16}$ V1model architecture¶¶. Along with these key parameters the mirrored packet carries the flow identifier and the switch identifier to the Telemetry collector. The telemetry collector then computes some additional parameters from the INT data of the two switches such as *packet loss* and *jitter*.

As an alternate to passive INT monitoring, we also leverage active probes in the network system. We have two such probes, one at the eNodeB and other at the SPGW-U. The first one measures delay and packet loss on S1-U interface, and the other one at the SGi interface by injecting Ping [8] messages from eNodeB and SPGW-U, destined to the SPGW-U and the test server, respectively.

Lastly to track the resource utilization in the mobile network, we monitor load on the CPU, memory and I/O disk operations with Sysstat. In current architecture, we exploit Sysstat parameters collected at the SPGW-U and the FlexRAN controller. These are the network components where we introduce our test bottleneck scenarios.

4.2 Bottleneck Detection

We formulate the bottleneck detection stage as anomaly detection problem for multivariate time series. We denote the measurements used to identify bottlenecks as multivariate time series $T = \{x_1, .., x_T\}$, $x(t) \in \mathbb{R}^m$ is an m-dimensional vector of samples for m variables at timestamp t. The goal of anomaly detection methods is to learn a model to label a binary variable $y_t \in \{0, 1\}$ at time t if anomaly is detected when it experiences an unseen observation $x_t^{'}$. As it is not easy to collect training data for all types of bottlenecks, it is therefore important to use an anomaly detection model that can learn in an unsupervised fashion to flag any type of bottlenecks as anomaly. Unsupervised anomaly detection is assuming that T contains only normal samples and the model is trained to learn the distribution of normal data. An anomalous point is one that differs significantly from T. The difference between the sample $x_t^{'}$ and the normal data T is measured by an anomaly score, which is then compared to a threshold. If the score is above the threshold the point is considered as an anomaly.

4.3 Bottleneck Classification

Once a bottleneck has been detected by the previous stage, we need to specify the type and location of this bottleneck. We take it as a classification problem and formulate it as a supervised learning, where the model is trained with known class labels. More specifically, we define 10 classes of *single* bottlenecks (See Table 2). These are bottlenecks that have a signal source of occurrence e.g. congestion at S1-U. Besides the single bottlenecks, our classification model should also be able to classify the *composite* bottlenecks, that has more than sources e.g. data congestion at S1-U and stress on network resources. Additionally, the model should identify any bottleneck that is not experienced before as *unidentified*, instead of misclassifying them. Those new bottlenecks are registered in a log file along with their corresponding measurements features. If their occurrence increases, they can be labelled and used for retraining the classification model. To this end, new type of bottleneck is introduced in the bottleneck identification system.

5 SYSTEM IMPLEMENTATION

5.1 Types of Bottlenecks

To create bottlenecks at different parts of the network, we generate

(1) congestion on network data paths,
(2) introduce packet loss in the network
(3) overload network resources, and
(4) create interference at the radio access link.

Table 2 provides the complete list of bottlenecks that we test on our system architecture. To emulate **congestion**, we introduce an additional TCP traffic flow in downlink direction at the maximum bandwidth of the network link(s), using the iPerf*** tool. Next, we induce **packet loss** through the Linux traffic control feature NetEm †††. These experiments consist of either a high loss percentage of around 5% or a low loss percentage of 1%. This adversely effects the outgoing and incoming data flow. Thirdly, to **overload the network resources**, we stress out the CPU and memory resources with increase in input/output disk operations with the *stress-ng* tool [20]‡‡‡. Lastly, to create **radio interference** we deploy a GNU Radio§§§ noise source on a separate system using a

¶¶Standard Metadata: ttps://github.com/p4lang/behavioral-model/blob/main/docs/simple_switch.md

***iPerf: https://iperf.fr.

†††NetEm: https://wiki.linuxfoundation.org/networking/netem.

‡‡‡For high stress we use: `stress-ng –cpu 4 -d 1 –hdd-bytes 1G -m 1 –vm-bytes 1G –iomix 1 –iomix-bytes 1G`.
For low stress, we use: `stress-ng –cpu 1 -d 1 –hdd-bytes 256M -m 1 –vm-bytes 256M –iomix 1 –iomix-bytes 256M`.

§§§GNU Radio: https://www.gnuradio.org.

Type	Location	Measurements	Complexity
Congestion	S1-U	Downlink TCP iPerf at maximum available bandwidth	Single
Congestion	SGi	Downlink TCP iperf at maximum available bandwidth	
Congestion	S1-U and SGi	Downlink TCP iperf at maximum available bandwidth	
High stress on resources	SPGW-U	High CPU, Memory and Disk I/O stress	
High Stress on Resources	FlexRAN Controller	High CPU, Memory and Disk I/O stress	
Low packet loss	SPGW-U	1% packet loss	
High packet loss	SPGW-U	5% packet loss	
Radio interference	Radio access link	Radio frequency interference caused by a transmitter on the same frequency at which the test UE received data	
Low resource stress	SPGW-U	Low CPU, Memory and Disk I/O stress	
Low resource stress	FlexRAN Controller	Low CPU, Memory and Disk I/O stress	
High stress on resources	SPGW-U and FlexRAN Controller	High CPU, Memory and Disk I/O stress	Composite
Congestion, High resource stress	S1-U, FlexRAN Controller	Downlink TCP iPerf at maximum available bandwidth, High CPU, Memory and Disk I/O stress	
Congestion, High packet loss	S1-U, SPGW-U	Downlink TCP iPerfat maximum available bandwidth, 5% packet loss	
High packet loss, High resource stress	SPGW-U	5% packet loss, High CPU, Memory and Disk I/O stress	

Table 2: Types of bottlenecks tested on the test bed.

dedicated SDR Ettus USRP B210. The noise source generates an additive white Gaussian noise (AWGN) signal which has central frequency similar to that of the eNB radio carrier.

As indicated by Table 2 the test bottlenecks can be categorised into *single* and *composite* groups depending upon their complexity. *Single* bottleneck is caused by performance degradation at a single source and is of a single type. *Composite*, on the other hand is a result of either more than one type of performance degradation or is sourced by more than one locations, simultaneously.

5.2 Bottleneck Identification Framework

In our bottleneck detection unit, we use a variant of the well known anomaly detection architecture "autoencoder", more specifically variational autoencoder (VAE). An autoencoder (AE) [32] is an unsupervised artificial neural network composed of an encoder and a decoder. The encoder (in Equation 1) takes the input x and maps it into a set of latent variables z, whereas the decoder maps the latent variables z back into the input space as a reconstruction $x^{'}$ (Equation 2). W and b are the weight and bias of the neural network and σ is the nonlinear transformation function.

$$z = \sigma(W_{xh}x + b_{xh}) \tag{1}$$

$$x^{'} = \sigma(W_{xh}z + b_{xh}) \tag{2}$$

The difference between the original input vector X and the reconstruction $x^{'}$ is the reconstruction error as in Equation 3. An autoencoder learns to minimize this reconstruction error (loss).

$$loss = ||x - x^{'}|| \tag{3}$$

To avoid the over-fitting that may result from decoding the latent space z without any reconstruction loss, we use VAE. Instead of encoding an input as a single point, VAE encode input as a distribution over z. A sample point from this distribution is then decoded and the reconstruction error can be computed. Thus the encoders and decoders of VAE are called as probabilistic encoders and decoders. Besides the reconstruction error, the loss function of VAE has to regularise the latent variable z that can be done using Kulback-Leibler divergence (KL). The loss function can be expressed in Equation 4, where μ, λ are the mean and covariance of the distribution and N is the Gaussian distribution.

$$loss = ||x - x^{'}|| + KL[N(\mu_x, \lambda_x), N(0, 1)] \tag{4}$$

After training, VAE will reconstruct normal data very well, while failing to do so with anomaly data which the VAE has

not encountered.VAE uses the reconstruction probability as the anomaly score.

In our implementation, VAE encoder and decoder are both a a two hidden layer with 28, 14 dimensions for the first and second hidden layer, respectively. The latent variable is 7 dimensions. Parameters of VAE are estimated using cross-validation.

For bottleneck classification stage, the model should solve all the three types of classification problems:

(1) Multi-class, to predict one of the 10 single bottleneck classes;
(2) Multi-label classification, for predicting composite bottlenecks.
(3) unidentified classes; to predict new types of bottlenecks that are not introduced in the training.

To accomplish this task, we use a Multi-Layer Perceptron (MLP), a neural network with fully connected neurons among layers. MLP model has the capabilities to approximate, through supervised learning, the function that relates the input with the output. Our MLP classifier has an input layer that expects 48 inputs and three hidden layers with 64, 24, 16 neurons respectively (that is chosen with trial and error method). We use the rectified linear unit (ReLU) as an activation function in the hidden layers which converges very quickly during the training. The output layer matches the number of single bottleneck classes. Each node in the output layer has a sigmoid activation, which predicts a probability of class membership for the label, a value between 0 and 1. This means it will predict 10 probabilities for each input sample. These can be converted to class labels by rounding the values to either 0 or 1. The output probability indicates the confidence of the classifier in its predictions. Based on this we can identify all types of bottlenecks. For instance, an input sample with the output [0.1, 0.98, 0.2, 0.4, 0.09, 0.08, 0.3,0.1,0.03,0.12] most likely belongs to the second bottleneck class. For the composite bottleneck, more than one class will have high probability above 0.5 threshold, e.g. [0.1, 0.09, 0.89, 0.78, 0.19, 0.31, 0.17, 0.42,0.08,0.13]. This way, any sample that does not belong to the any of the bottleneck classes can me marked as unidentified. For example if the output of the MLP for a sample is [0.21, 0.13, 0.4, 0.2, 0.09, 0.15, 0.32, 0.3,0.07,0.42] it would not belong to any of the learned classes and would be "unidentified". The model is fitted using Adam optimiser [21] with learning rate 0.001, and adopting cross-entropy as loss function over 200 training epochs. After model training, in the classification phase, the classifier makes decisions on each input sample and determines the bottleneck class that each sample (i.e. feature vector) belongs to. The decision is based on the probability generated by the output layer.

6 EVALUATION

6.1 Dataset

To evaluate our proposed system, we exploit three different datasets based on the design choices described above.

6.1.1 UE-based Dataset. It comprises the measurements collected by the UE (see Table 1). In the UE dataset, 8 features are measured directly from UE namely, *AvgRTT*, *Packet loss percentage*, *Transfer(MB)*, *Bitrate(Mbps)*,*Jitter(ms)*, *RSRP*, *RSRQ*, and *RSSNR*. *Inter packet gap (IPG)* is a metric computed from the monitored feature of *AvgRTT*. Our monitoring frequency is every second, but for data analytics we take mean values of the features in each 5 seconds window. For each of these windows we compute other statistical metrics including median, skewness and kurtosis of all the primary features given above, except of the radio quality indicators. The resulting UE dataset consists of 28 features that we use for bottleneck detection. Within this dataset 8,640 data samples were collected under normal network conditions, i.e baseline. In the context of this study a data sample denotes set of features that were derived from a single 5 seconds window. 40,320 data samples were collected when various bottlenecks were emulated in the network.

6.1.2 Mobile Network Dataset. It contains the measurements collected from the components of mobile network, namely, eNB, FlexRAN, SPGW-U and P4 switches. These measurements were reported to the telemetry collector shown in Table 1. Pre-processing steps are applied to the measurements before feeding into the bottleneck identification model. Initially, we filtered out all the missing values and constant features. In particular, we remove all configurations and identifiers variables such as *cell_config.init_nr*, *cell_config._cell_id*, *ue_config.rnti* and *imsi* from the FlexRAN measurements. Secondly we retained only one of the correlated features from single monitoring point, for example, retaining *kbmemused* and removing its correlated feature of *kbavail* from memory usage features reported by *Sysstat* at SPGW-U. The switches at S1-U and SGi perform INT and mirrors the modified packets to the telemetry controller, which not only retrieved the INT features defined in the Table 1 but also computed metrics i.e., percentage of lost packets and uplink & downlink jitter between INT packets coming from the two P4 switches. Lastly for the measurements that we collected frequently, that is every second, their statistical metrics including mean, median, kurtosis and skewness were computed for every 5 seconds window. This resulted in a dataset consists of 76 features. Just like user dataset, the final dataset from mobile network consist of 48,960 data samples.

6.1.3 Network-wide Dataset. It combines all features from both UE-based dataset and network dataset. It is used for

Table 3: Performance of our proposed bottleneck identification system using different types of measurements.

Dataset	Bottleneck detection			Bottleneck classification		
	P	R	F1	P	R	F1
UE-based	0.90	0.81	0.85	0.62	0.54	0.58
Mobile network-based	0.92	0.87	0.89	0.90	0.88	0.89
Network-wide	0.94	0.89	0.91	0.95	0.91	0.93

investigating the trade-off between the centralized and decentralized system architechures.

6.2 Evaluation Metrics

To evaluate the performance of our bottleneck identification framework, we use prediction (P), recall (R) and F1-score (F1) [16]:

$$P = \frac{TP}{TP + FP}, \qquad R = \frac{TP}{TP + FN}, \qquad F1 = 2 * \frac{P * R}{P + R},$$

where TP is True Positives, FP is False Positives and FN is False Negatives.

We formulate the composite bottlenecks as multi-label classes, and use the following metrics to evaluate their classification accuracy:

- Hamming Loss: It is equal to the number of incorrect predicted labels (TNIP) divided by the total number of predictions (TNP). In hamming loss the smaller the result, the better is the model.

$$Hamming\,Loss = \frac{TNIP}{TNP}$$

- Exact Match Ratio (EMR): is the most strict metric, indicating the percentage of samples that have all their labels classified correctly. In our case, the two single bottlenecks that form the composite have to be classified correctly in order to be considered as TP. The disadvantage of this measure is that multi-class classification problems have a chance of being partially correct, but here we ignore those partially correct matches.

In the bottleneck detection, we define the threshold for anomaly score that provides the best F1 score.

6.3 UE-based vs. Mobile Network-based Bottleneck Identification

First, we evaluate the performance of our proposed bottleneck identification system based on UE measurements, mobile network measurments and network-wide measurements. For the bottleneck detection, we use our VAE model defined in subsection 5.2. We train the model with baseline measurements and test them using a mix of baseline and bottleneck measurements. As shown in Table 3, VAE demonstrates relatively excellent bottleneck detection capability and achieves 0.85, 0.89, and 0.91 F1 scores on UE-based dataset, mobile network-based dataset and network-wide dataset, respectively. Using network-wide measurements to detect bottlenecks results in higher performance (7% F1-score improvement) compared when using only UE-measurements. This is because some types of bottlenecks such as low stress and low loss at FlexRAN and SPGW-U are hardly being identified by UE as shown in Figure 3. This is also confirmed by the distributions of UE measurements in Figure 4. For example, UE measurements (*Average RTT*, *Bitrate(Mbps)*, *Packet loss percentage* and *RSSNR*) of low stress bottleneck at FlexRAN and SPGWU have similar distribution to the measurements of the baseline. Due to the same reason, the classification model based on UE measurements shows very poor performance of 58% F1-score compared with the network-based and network-wide datasets. However, still UE measurements demonstrate significant contribution in identifying other types of bottlenecks. For example, bottleneck caused by radio interference can be easily flagged by the monitored *RSSNR* by the UE. Based on our proposed architecture in Figure 2, our classification model can accurately identify and attribute 92% of the bottlenecks when relying on network dataset.

Takeaways. Leveraging measurements at UE can help in detecting bottlenecks with relatively good performance. The bottleneck with low stress on the resources are, however, exceedingly difficult to detect using solely UE measurements. The stress on the resources is low enough to adversely impact performance at user end. It needs measurements from rest of the mobile network to identify and attribute both the high and low impact bottlenecks.

6.4 Single and composite bottlenecks identification.

Here, we dig deeper in understanding the performance of our system on classifying single and composite bottlenecks based on network-based measurements. Table 4 presents the evaluation results per bottleneck type for single bottlenecks. As shown in this table, based on mobile network-based dataset our model is able to classify the defined bottlenecks with high accuracy; F1 score is above 0.84 and in most types above 0.9. A closer look reveals that the model cannot distinguish the bottlenecks caused by loss at SPGW-U with each other; i.e low and high losses resulting in more FNs than the other types. Furthermore, the bottleneck due to radio interference is showing lower value of 0.81 recall.

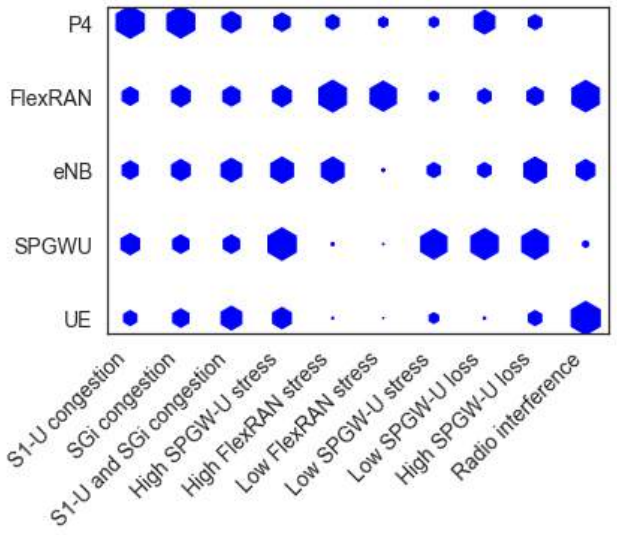


Figure 3: Contribution of measurements from different points in the network in bottleneck identification using SHAP framework [1]. The higher contribution of measurements is made by larger hexagons.

Table 4: Performance evaluation of single bottleneck classification.

Single Bottlenecks			
Bottleneck class	**P**	**R**	**F1**
S1-U congestion	0.94	0.88	0.91
SGi congestion	0.96	0.90	0.88
S1-U and SGi cong.	0.90	0.86	0.85
High SPGW-U stress	0.97	0.88	0.93
Low SPGW-U stress	0.87	0.84	0.85
High FlexRAN stress	0.96	0.90	0.93
Low FlexRAN stress	0.85	0.82	0.83
Low SPGW-U loss	0.85	0.83	0.84
High SPGW-U loss	0.86	0.88	0.87
Radio interference	0.89	0.81	0.84

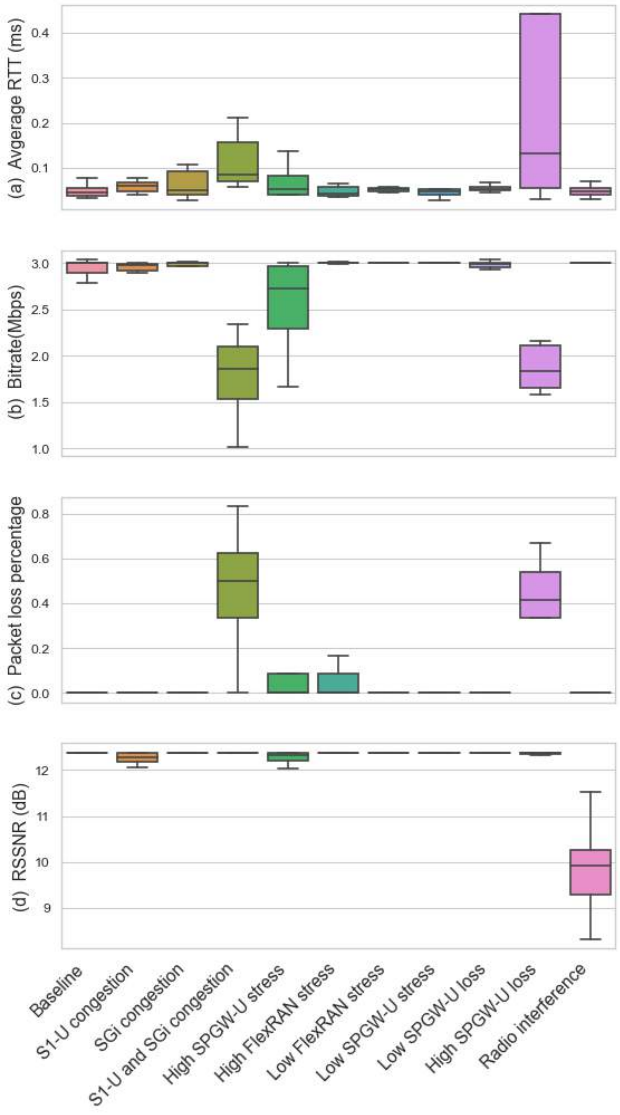


Figure 4: Distribution of most important UE features collected under different test scenarios.

Both Figure 3 and Figure 4 show that bottleneck generated by radio interference is mainly identified by UE measurements, more specifically *RSSNR* metric. Features from FlexRAN and eNB measurements too have partial contribution in identifying bottlenecks caused by interference, namely, *pdcp_stats.pkt_tx_bytes* and *.wb_cqi*

Next we evaluate the performance of composite bottlenecks classification. These results are summarized in Table 5 using the EMR method and the hamming loss. Although, our classifier shows superior performance in single bottlenecks prediction, the performance is degraded in the case of the composite. This is mainly due to our assumption that there is no correlation between the single bottlenecks that form the composite bottleneck. However, the hamming loss is still showing that our model is able to detect high proportion of the single bottlenecks that form the composite bottlenecks. For example, in the case of SPGW-U stress/ FlexRAN stress the hamming loss is 0.27 which means if there are 100 samples for composite bottlenecks, our the model will predict incorrectly about 27% of the individual single bottleneck classes. Furthermore, based on EMR metric we can observe that the performance of the model in detecting composite

Table 5: Performance evaluation of composite bottleneck classification.

Composite Bottlenecks				
Bottleneck classes	**P_{EMR}**	**R_{EMR}**	**$F1_{EMR}$**	**Hamming loss**
SPGW-U stress, FlexRAN stress	0.84	0.83	0.83	0.27
S1-U congestion, FlexRAN stress	0.82	0.79	0.80	0.23
S1-U congestion, High SPGW-U loss	0.80	0.77	0.78	0.20
SPGW-U stress, High SPGW-U loss	0.77	0.71	0.73	0.25

bottleneck in the same location is marginally worse than the case of different locations bottlenecks. For instance, {*SPGW-U stress, High SPGW-U loss*} exhibits lower $F1_{EMR}$ which means the model hardly can identify both bottlenecks, however it still shows low hamming loss which indicates that the model is able to identify at least one of the bottlenecks most of the time.

6.5 Classification of unidentified bottlenecks

In this experiment, we evaluate the efficiency of our model in case of unidentified bottlenecks. We train our model using a dataset labelled with 9 types of single bottlenecks from Table 2. For evaluation we introduce the unidentified bottleneck along with others being used in the training. Figure 5 shows the average probabilities of classifying unidentified bottlenecks per labelled bottlenecks. For example, if the unidentified bottleneck is *S1-U congestion*, our model predicts it as *SGi congestion* with a probability of 0.21 which is rounded to 0. In case of all probabilities below 0.5, the bottleneck is marked as *unidentified*. Bottlenecks such as *S1-U congestion*, *SGi congestion* and *radio interference* are classified as unidentified bottlenecks with high accuracy ($\approx$ 98%) if they are not introduced into our model during training. On the other hand, the model can identify the type and the location of the bottleneck if it experiences the same bottleneck during training however with different severity. In our case, if the model is trained with high stress at SPGW-U and tested against low stress at SPGW-U, our model classifies it as high stress with a confidence of 63%. This also applies to other bottlenecks with different severity such as stress at FlexRAN and loss at SPGW-U.

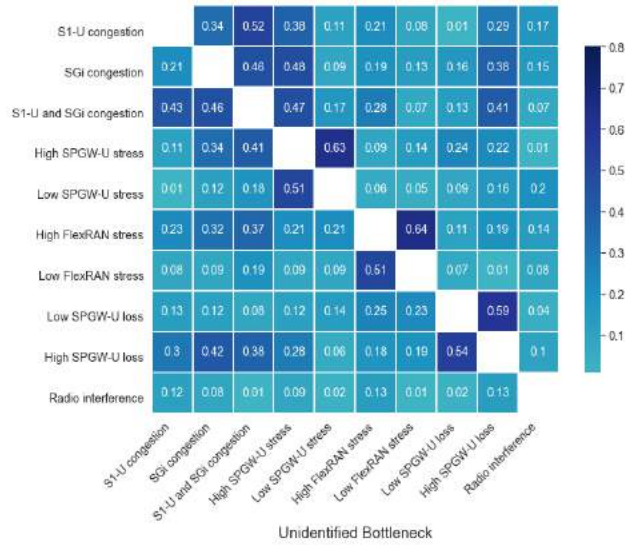


Figure 5: Average probability of each bottleneck classes computed by the classification model in case of unidentified bottlenecks.

6.6 Active vs. P4 measurements

In our measurement dataset from mobile network the characteristics of user flows on S1-U and SGi interface are monitored in two ways i.e. by active monitoring and by P4 based INT monitoring. Since active monitoring injects additional load on the network links by interfering with the user data, in this experiment we investigate the potential of P4 based telemetry in replacing active monitoring.

Figure 6 depicts the classification accuracy when measurement features from either active monitoring or P4 based telemetry are used with rest of the measurement features. When utilizing P4 based INT features, accuracy across the various types of bottlenecks is comparable to that obtained from active features.

7 DISCUSSION AND FUTURE WORKS

In this section we evaluate the overhead associated with our distributed telemetry framework. We asses the framework from the perspective of:

- Traffic overhead on the network links
- CPU and memory overhead on the cloudified mobile network
- Computational overhead of the data analytics system

Telemetry comes with processing, memory and bandwidth cost of different intensities. Passive monitoring only induce bandwidth cost when measurements are sent to an central entity such as telemetry collector for data analytics. Compared to passive monitoring, active monitoring places additional

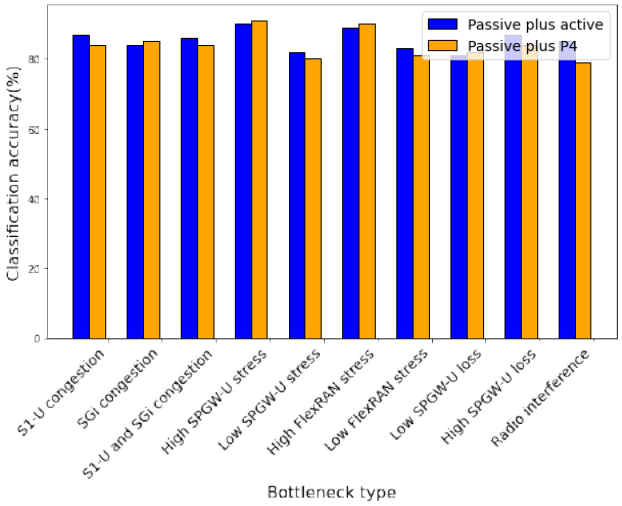


Figure 6: Trade-off between active measurements and P4 measurements.

overhead on network links by interfering with the actual user traffic. In both cases less frequent and sporadic monitoring is advised to reduce the cost []. Reduction of monitoring frequency, however misses short lived anomalies. Our framework, therefore leverages frequent monitoring at user end. Once an anomaly is detected, the UE can then trigger cloudified mobile network to enhance it monitoring frequency so to localize source of performance degradation in case it lies within its domain. To minimize processing and memory overhead on the UE, the crowdsourced network coverage and ping tests along with passive monitoring features of user applications, can be off-loaded to an edge computing device.

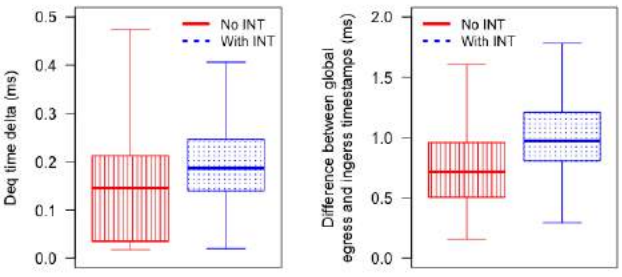


Figure 8: Impact on data packets when a virtual P4 switch performs in-band network telemetry in a mobile network.

8 RELATED WORK

There is big interest in developing telemetry solutions for Software-Defined Networking (SDN) based networks [4, 18, 26]. Advanced monitoring solutions have been proposed to identify bottlenecks, while attempting to balance high detection rates with minimal monitoring overhead costs.

z-TORCH [33] is an automated Network Function Virtualization (NFV) orchestration solution utilizing Machine Learning (ML) techniques to enhance quality of decisions in Management and Orchestration (MANO) systems. It adapts monitoring load based on Virtual Network Function (VNF) profile time variations. CellScope [15] uses domain specific knowledge to apply Multi-task Learning (i.e. use of several models in parallel) on Radio Access Network (RAN) data. They are able to perform RAN troubleshooting and mobile phone energy bug diagnosis more efficiently compared to contemporary solutions. The authors of [35] propose self-tuning, adaptive monitoring mechanism that adjusts measurement granularity based on observed traffic dynamics.

In-band Telemetry for Softwarised Networks. In the context of mobile core networks, P4 switches have been used for real-time attack detection and mitigation [2], enhancing the User Plane Function (UPF) functionality [22] and ensuring Quality-of-Service (QoS) at the slice level [30]. LossSight [34] tackles the problem of packet loss when using Inband Network Telemetry (INT), through "Alternate Marking" the telemetry headers of each flow. They are able to correctly identify and locate packet loss events, even when telemetry information is lost alongside the packets carrying it. There are also approaches that do not rely on P4. The authors of [17] propose an extension to INT tuned for wireless multihop networks. Each node runs both a telemetry module and an agent that defines the optimal local traffic engineering policy.

AI Processing of Telemetry Data. To troubleshoot network with anomalies in performance, knowledge about the causes of anomalies is the necessary. [13] presents an Artificial Intelligence (AI)-powered trustworthy distributed Self driving network framework. The framework responds to P4 telemetry data and is able to modify the P4 code based on detected events. M. Moulay et al. [23] propose an unsupervised ML method, Troubleshoot Trees (TTrees), to classify the reason of an anomaly in cellular performance with minimal amount of data and quick training. For a set of Key Performance Indicators (KPIs), TTree first uses a ML method to separate instances that cannot be classified. Then it uses a clustering method to group these unclassified instances, that are considered anomalies, into separate groups. At this latter stage an expert should inspect the clusters and assign that meaning [12].

Telemetry focused testbeds. Similar OpenAirInterface (OAI)-based testbeds have been used to study applications of network telemetry. The authors of [27] identify a range of

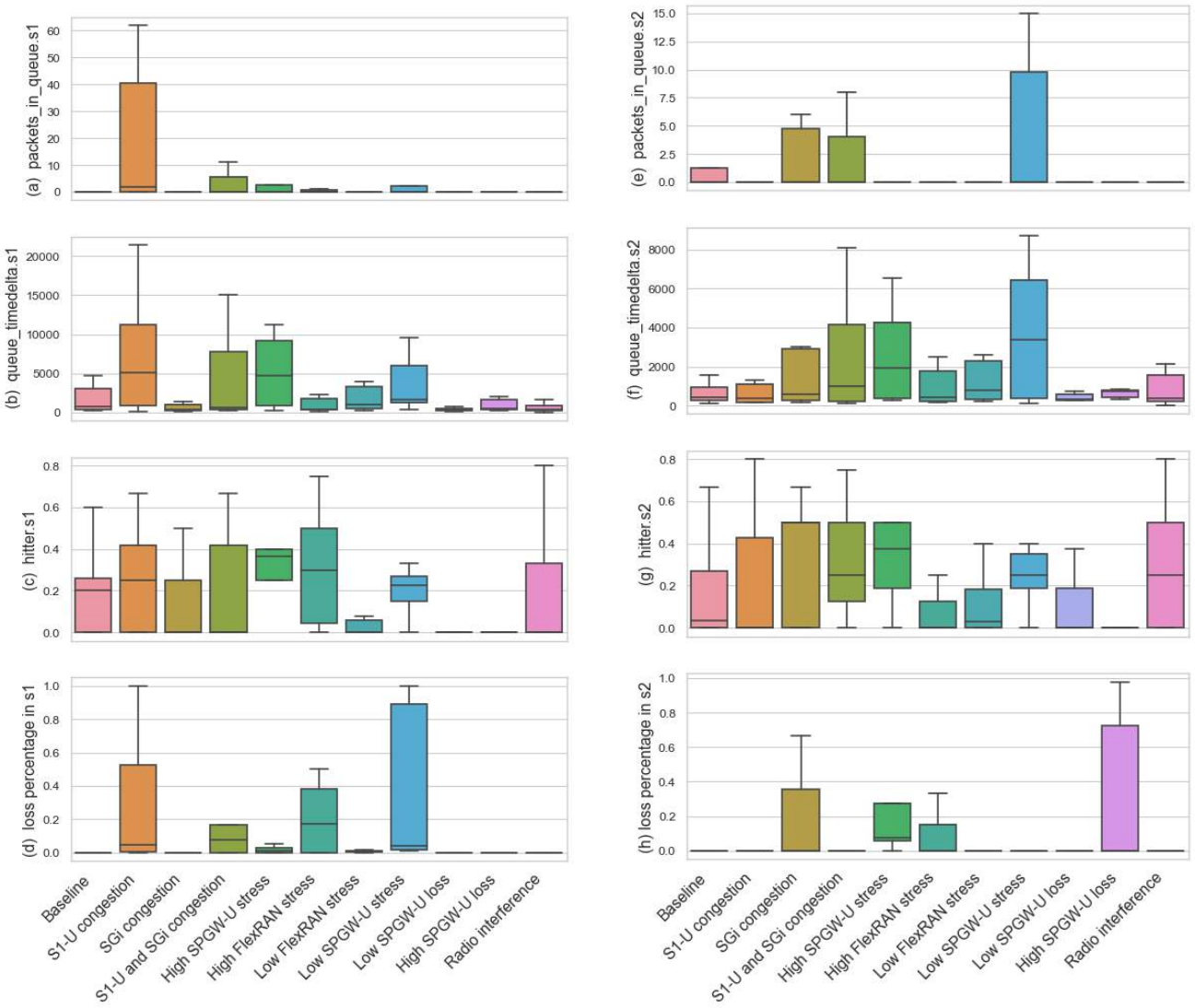


Figure 7: Distribution of P4 measurements collected under different experiments.

performance bottlenecks that 5G networks experience, by gathering monitoring information at several layers of each network component. Then they use machine learning to map data sources to performance bottlenecks. [37] proposes an exposed Monitor-Analyze-Plan-Execute with Knowledge (MAPE-K) closed-loop model to automate Service Assurance, across slices. It exposes monitoring information to slice owners who are then able to provide their insights to the Analyze module of the closed loop, enabling a more robust virtualized network infrastructure. In contrast to the above, we enhance the data collection with INT data provided by virtual P4 switches.

9 CONCLUSION

5G network has transformed the mobile network into a multi-service architecture that supports diverse use cases with varying requirements and needs to ensure that the Service Level Agreements (SLA) of customized end-to-end sliced services are met. This requires monitoring the resource usage and characteristics of data flows at the virtualized network components and interfaces of its cloud mobile network, as well as tracking the performance of its radio interfaces and UEs. We implement our proposed telemetry architecture on a software defined virtualised testbed that resembles a cloudified mobile network. Further, monitored data are

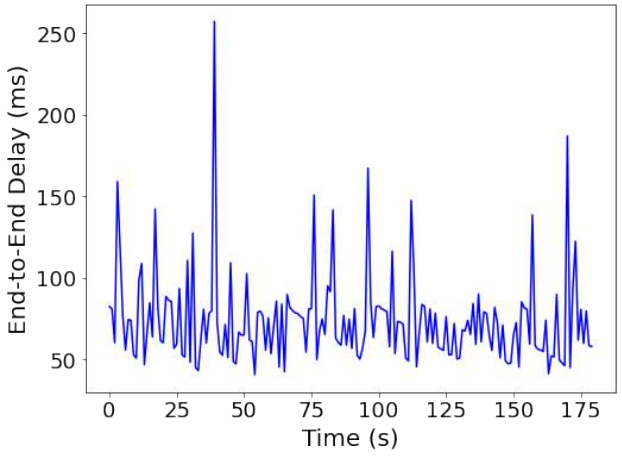


Figure 9: An example of end-to-end performance data from mobile broadband network. The time series spans 3 minutes and have one second granularity. A non-frequent monitoring may not trigger a performance issue, until it deteriorates

used to evaluate our bottleneck identification framework. Although the centralized architecture which combines all measurements from UE and mobile network (i.e network-wide dataset) shows superior performance results in both bottleneck detection and classification, our distributed framework still demonstrates comparable accuracy while keeping system overhead minimal.

REFERENCES

[1] Liat Antwarg, Ronnie Mindlin Miller, Bracha Shapira, and Lior Rokach. 2019. Explaining anomalies detected by autoencoders using SHAP. *arXiv preprint arXiv:1903.02407* (2019).

[2] Michel Bonfim, Marcelo Santos, Kelvin Dias, and Stenio Fernandes. 2020. A real-time attack defense framework for 5G network slicing. *Software: Practice and Experience* 50, 7 (2020), 1228–1257.

[3] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, et al. 2014. P4: Programming protocol-independent packet processors. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014), 87–95.

[4] Raouf Boutaba, Nashid Shahriar, Mohammad A Salahuddin, Shihabur R Chowdhury, Niloy Saha, and Alexander James. 2021. AI-driven Closed-loop Automation in 5G and beyond Mobile Networks. In *Proceedings of the 4th FlexNets Workshop on Flexible Networks Artificial Intelligence Supported Network Flexibility and Agility*. 1–6.

[5] Mihai Budiu and Chris Dodd. 2017. The P4-16 Programming Language. *ACM SIGOPS Operating Systems Review* 51, 1 (2017), 5–14.

[6] Thomas Dreibholz. 2020. A 4G/5G Packet Core as VNF with Open Source MANO and OpenAirInterface. In *Proceedings of the 28th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. Hvar, Dalmacija/Croatia, 3 pages. https://doi.org/10.23919/SoftCOM50211.2020.9238222

[7] Thomas Dreibholz. 2020. Flexible 4G/5G Testbed Setup for Mobile Edge Computing using OpenAirInterface and Open Source MANO. In *Proceedings of the 2nd International Workshop on Recent Advances for Multi-Clouds and Mobile Edge Computing (M2EC) in conjunction with the 34th International Conference on Advanced Information Networking and Applications (AINA)*. Caserta, Campania/Italy, 1143–1153. https://doi.org/10.1007/978-3-030-44038-1_105

[8] Thomas Dreibholz. 2020. HiPerConTracer - A Versatile Tool for IP Connectivity Tracing in Multi-Path Setups. In *Proceedings of the 28th IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM)*. Hvar, Dalmacija/Croatia, 6 pages. https://doi.org/10.23919/SoftCOM50211.2020.9238278

[9] Mah-Rukh Fida, Andres F. Ocampo, and Ahmed Elmokashfi. 2022. Measuring and Localising Congestion in Mobile Broadband Networks. *IEEE Transactions on Network and Service Management* 19, 1 (2022), 366–380. https://doi.org/10.1109/TNSM.2021.3115722

[10] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. 2017. Network slicing in 5G: Survey and Challenges. *IEEE Communications Magazine* 55, 5 (2017), 94–100.

[11] Sébastien Godard. 2022. Sysstat Tutorial. http://sebastien.godard.pagesperso-orange.fr/tutorial.html

[12] TPAW Group et al. 2020. In-band Network Telemetry (INT) data plane specification.

[13] Othmane Hireche, Chafika Benzaïd, and Tarik Taleb. 2022. Deep data plane programming and AI for zero-trust self-driven networking in beyond 5G. *Computer Networks* 203 (2022), 108668.

[14] Matthias Hirth, Tobias Hossfeld, Marco Mellia, Christian Schwartz, and Frank Lehrieder. 2015. Crowdsourced network measurements: Benefits and best practices. *Computer Networks* 90 (07 2015). https://doi.org/10.1016/j.comnet.2015.07.003

[15] Anand Padmanabha Iyer, Li Erran Li, Mosharaf Chowdhury, and Ion Stoica. 2018. Mitigating the Latency-Accuracy Trade-off in Mobile Data Analytics Systems. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (New Delhi, India) *(MobiCom '18)*. ACM, New York, NY, USA, 513–528. https://doi.org/10.1145/3241539.3241581

[16] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.

[17] Prabhu Janakaraj, Pinyarash Pinyoanuntapong, Pu Wang, and Minwoo Lee. 2020. Towards in-band telemetry for self driving wireless networks. In *IEEE INFOCOM 2020-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 766–773.

[18] Grigorios Kakkavas, Adamantia Stamou, Vasileios Karyotis, and Symeon Papavassiliou. 2021. Network tomography for efficient monitoring in SDN-enabled 5G networks and beyond: Challenges and opportunities. *IEEE Communications Magazine* 59, 3 (2021), 70–76.

[19] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J. Wobker. 2015. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*, Vol. 15. 2 pages.

[20] Colin Ian King. 2022. stress-ng. https://wiki.ubuntu.com/Kernel/Reference/stress-ng

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Robert MacDavid, Carmelo Cascone, Pingping Lin, Badhrinath Padmanabhan, Ajay Thakur, Larry Peterson, Jennifer Rexford, and Oguz Sunay. 2021. A P4-based 5G User Plane Function. In *Proceedings of the ACM SIGCOMM Symposium on SDN Research (SOSR)*. 162–168.

[23] Mohamed Moulay, Rafael García, Vincenzo Mancuso, Pablo Rojo, and Antonio Fernández Anta. 2021. TTrees: Automated Classification of Causes of Network Anomalies with Little Data. In *22nd IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2021)*.

[24] OpenAirInterface. 2019. *How to Connect OAI eNB (USRP B210) with COTS UE.*

[25] OpenStack. 2020. *OpenStack Installation Guide.*

[26] Francesco Paolucci, Filippo Cugini, Piero Castoldi, and Tomasz Osiński. 2021. Enhancing 5G SDN/NFV edge with P4 data plane programmability. *IEEE Network* 35, 3 (2021), 154–160.

[27] Georgios Patounas, Xenofon Foukas, Ahmed Elmokashfi, and Mahesh K. Marina. 2020. Characterization and Identification of Cloudified Mobile Network Performance Bottlenecks. *IEEE Transactions on Network and Service Management* 17, 4 (2020), 2567–2583. https://doi.org/10.1109/TNSM.2020.3018538

[28] Jonathan Bruce Postel. 1981. *Internet Protocol.* RFC 791. IETF. https://doi.org/10.17487/RFC0791

[29] Andy Reid, Andrés González, Antonio Elizondo Armengol, Gerardo García de Blas, Min Xie, Pål Grønsund, Peter Willis, Phil Eardley, and Francisco-Javier Ramón Salguero. 2019. *OSM Scope, Functionality, Operation and Integration Guidelines.* White Paper. ETSI.

[30] Ruben Ricart-Sanchez, Pedro Malagon, Antonio Matencio-Escolar, Jose M Alcaraz Calero, and Qi Wang. 2020. Toward hardware-accelerated QoS-aware 5G network slicing based on data plane programmability. *Transactions on Emerging Telecommunications Technologies* 31, 1 (2020), e3726.

[31] Mohammad Azmi Ridwan, Nurul Asyikin Mohamed Radzi, Fairuz Abdullah, and YE Jalil. 2021. Applications of machine learning in networking: A survey of current issues and future challenges. *IEEE Access* 9 (2021), 52523–52556.

[32] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1985. *Learning internal representations by error propagation.* Technical Report. California Univ San Diego La Jolla Inst for Cognitive Science.

[33] Vincenzo Sciancalepore et al. 2018. z-TORCH: An Automated NFV Orchestration and Monitoring Solution. *IEEE Transactions on Network and Service Management* (2018).

[34] Lizhuang Tan, Wei Su, Wei Zhang, Huiling Shi, Jingying Miao, and Pilar Manzanares-Lopez. 2021. A packet loss monitoring system for in-band network telemetry: Detection, localization, diagnosis and recovery. *IEEE Transactions on Network and Service Management* 18, 4 (2021), 4151–4168.

[35] Gioacchino Tangari, Daphne Tuncer, Marinos Charalambides, Yuanshunle Qi, and George Pavlou. 2018. Self-Adaptive Decentralized Monitoring in Software-Defined Networks. *IEEE Transactions on Network and Service Management* 15, 4 (2018), 1277–1291. https://doi.org/10.1109/TNSM.2018.2874813

[36] Min Xie, Thomas Dreibholz, Foivos Ioannis Michelinakis, Joan Pujol-Roig, Wint Yi Poe, Ahmed Mustafa Elmokashfi, Sayantini Majumdar, and Sara Malacarne. 2021. An Exposed Closed-Loop Model for Customer-Driven Service Assurance Automation. In *Proceedings of the 30th IEEE European Conference on Networks and Communications (EuCNC).* Porto/Portugal, 419–424. https://doi.org/10.1109/EuCNC/6GSummit51104.2021.9482533

[37] Min Xie, Foivos Michelinakis, Thomas Dreibholz, Joan S Pujol-Roig, Sara Malacarne, Sayantini Majumdar, Wint Yi Poe, and Ahmed M Elmokashfi. 2021. An Exposed Closed-Loop Model for Customer-Driven Service Assurance Automation. In *2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit).* IEEE, 419–424.

[38] Min Xie, Joan Sebastià Pujol-Roig, Foivos Ioannis Michelinakis, Thomas Dreibholz, Carmen Guerrero, Adrián Gallego Sánchez, Wint Yi Poe, Yue Wang, and Ahmed Mustafa Elmokashfi. 2020. AI-Driven Closed-Loop Service Assurance with Service Exposures. In *Proceedings of the 29th IEEE European Conference on Networks and Communications (EuCNC).* Dubrovnik, Dubrovnik-Neretva/Croatia, 265–270. https://doi.org/10.1109/EuCNC48522.2020.9200943

POSTADRESSE:
OsloMet – storbyuniversitetet
Pilestredet 46
Postboks 4, St. Olavs Plass
0130 Oslo

OsloMet Avhandling 2023 nr 23
ISSN 2535-471X
ISBN 978-82-8364-490-6