Universiteit
Antwerpen

University of Antwerp
Faculty of Science
Department of Mathematics &
Computer Science
Research group IDLab

Multi-technology Management of Heterogeneous Wireless Networks

Multi-technologie beheer van heterogene draadloze netwerken

Tom De Schepper

IDLab
INTERNET & DATA LAB

Submitted in fulfillment of the
requirements for the degree of
Doctor in Science: Computer Science
Academical year 2018 - 2019

University of Antwerp
Faculty of Science
Department of Mathematics &
Computer Science
Research group IDLab

University of Antwerp
Faculty of Science

Department of Mathematics & Computer Science
Sint-Pietersvliet 7, B-2000 Antwerp, Belgium

Submitted in fulfillment of the
requirements for the degree of
Doctor in Science: Computer Science
Academical year 2018 - 2019

Acknowledgments

In ancient Greek mythology, heroes, often demigods, had to prove that they could go the distance by completing a series of challenges or quests. Arguably, the most famous amongst these demigods is Heracles. He is, among others, known for slaying the nine-headed Hydra, defeating the Nemean Lion, and cleaning the Augean stables in a single day. Other heroes that achieved glory are, for example, Theseus, Perseus, and Jason.

Comparing a Ph.D. candidacy to a hero's quest in ancient Greek times could be considered bold, arrogant, and a perfect example of Hubris. It can be interpreted as nothing less than asking the twelve Olympians to either strike you down with a thunderbolt, feeding you to the Minotaur, or throwing you directly into Tartarus. However, I can't help but notice several strong resemblances. First of all, bearing the burden of a nearly impossible task, while your faith is often out of your own hands. Furthermore, the hurdles that need to be taken although being counteracted by all kinds of evil forces (like reviewers). Finally, also the sweet taste of success at the end of the four-year journey is similar to the great warm welcome that awaits a hero. Now, when writing the final sentences of this dissertation, I am proud and grateful that I am able to look back on the traveled journey and its experiences. Although a Ph.D. study is largely an individual crusade, I would not have been able to complete it without the support, patience, and advice of many people.

While Greek heroes were typically trained and mentored by Chiron the centaur, I had the privilege of being supervised by both prof. dr. Steven Latré and prof. dr. Jeroen Famaey. As such, they together definitely met the four-legged standard that was defined in ancient Greece. Throughout the past four years, I learned a lot from them and without a doubt, I would not be the researcher I am today if it wasn't for Steven and Jeroen. They both complemented each other nicely and, looking back, I could not have wished for better promoters. I appreciated very much the open communication style, the critical judgment, the ambitious goals, and the open door when needed. The non-hierarchical supervision style allowed for interesting and open-minded discussions, while there was also room for own ideas and input. This dissertation would not have been possible without their confidence, insights, and feedback. I am also grateful for getting the opportunity to pursue a Ph.D. title in the first place. While I originally took some time to provide a positive answer to the job position in 2015 and had some doubts along the road, I am happy to have done this Ph.D. study in the end. I would also like to explicitly thank both Steven and Jeroen for offering me an interesting position in their respective teams after finishing my Ph.D. candidacy.

Furthermore, I like to thank all members of the jury for being willing to read through this dissertation of over 200 pages and providing the additional insightful feedback that made it possible to finalize this manuscript. I have met all jury members at different times and locations the past four years throughout this Ph.D. study, or even before. While I have worked more closely together with some of them, I had interesting talks, discussions, and collaborations with all of them. On that note, I want to express my appreciation for the collaboration with Michael, in particular, for his efficient management style and constructive criticism, over the past year on the B-budget project.

Over the years, I had the opportunity to work together with a large number of different people, both from within or outside of the IDLab research group. Together, we conquered research challenges, made presentations, held brainstorm meetings, guided master theses or internships, constructed demos, conducted job interviews for future colleagues, and most importantly wrote papers, rewrote papers, and even re-rewrote papers. As such, all of them were involved in this Ph.D. research. I want to thank two people by name since I worked very closely with them over longer periods. First of all, Patrick, with whom I have shared all three offices that I have occupied in the past four years. Despite his profound love for cursing out loud, we worked successfully together on a very regular basis throughout the majority of the past four years. Second, I also want to mention Miguel, with whom I had a great collaboration over the last year where we worked together on the topic of traffic recognition.

In Greek mythology, the faith of all was bestowed upon the three Moirai as they controlled the thread of life for every mortal. As such, they were able to decide over life and death. Within our research group, this power resides in the hands of Lynn, Anne, and Hanne. On a daily basis, the three of them were always available for assisting in navigating the administrative and financial labyrinth, providing ice creams on hot summer days, handle orders, organize Easter egg hunts, etc. Without a doubt, they were a tremendous and well-appreciated help on many occasions, including finishing up my Ph.D. study. Furthermore, I really appreciated the willingness of Johan to help out with all kinds of questions or problems and Maxim for providing his template as a starting point for this Ph.D. book.

Overall, it has always been an exquisite pleasure of being a part of the IDLab research group in Antwerp. This is mainly due to the privilege of working together with an international band of wonderful, interesting, friendly, open-minded, and (sometimes) slightly odd colleagues. While I acknowledge that I often skipped coffee breaks or bar visits, I had many interesting talks, discussions, and laughs with most of you. In particular, I would like to thank Anne, Carmen, Hanne, Kathleen, Lynn, Paola, Serena, Bart (all four of them), Carlos, Chris, Daniel, Dimitri, Ensar, Esteban, Glenn, Jakob, Jeremy, Jeroen, Johan, Johann, Kevin, Le, Matthias, Maxim, Michael, Miguel, Niels, Patrick, Pedro, Ruben, Steven, Werner, Yorick, etc. Moreover, I have always welcomed the distractions of the intensive and very high standing ping pong games with Jakob, Ruben, and Matthias. I also enjoyed the attempts of speaking Spanish with Carmen, Maria, Carlos, Esteban, Johann, Miguel, or Paola and the conversations on tram 7 with Lynn.

The travels of Greek heroes were mostly situated in the Eastern Mediterranean region. In contrast, I had the luxury of visiting three different cities on three different continents. The trips to, respectively, Guadalajara, Tokyo, and Rome were absolutely memorable. My travel companions and the people that I met locally have strongly contributed to this. For this, I would like to thank Christian, Jakob, Steven, Anika, Maria, Ensar, José, Michael, and Patrick. Remarkable moments were, among others, the adventurous blitz visit of the center of Guadalajara, the tequila museum, the Senso-ji temple, the dazzling Robot Restaurant, meeting the Pope, and eating original Roman dishes a couple of meters underground. Moreover, it was not always necessary to leave the country for nice remarkable experiences. For instance, I will surely remember my participation in the science battle by presenting my research for primary school students or transporting a bag full of Lego on a tram during rush hour and building that Lego afterwards.

While pursuing a Ph.D. is a more than full-time occupancy, there is of course also a life outside of the Ph.D. research. For many people outside of the work environment, it is often unclear what a Ph.D. is all about and, even more, what the actual research is about. However, many people still supported me along the way of this four-year quest. Therefore, I want to thank my family and friends. First of all, I want to thank my parents who have always made sure that my brothers and I came in the first place, enabling the best possible future for the three of us. Without their efforts and choices, it would have been impossible to study at the university and, afterwards, pursue a Ph.D. I would also like to thank Chantal and Eric for letting me work in an asocial fashion behind my laptop for most weekends in the Ardennes. Furthermore, I have the luxury of having the best possible distraction to everyday life on the dance floor. A big thanks to Carla, Joeri, and the other *Yoly's* for the warm atmosphere and support on the dance floor, at the bar, and everywhere else in between those two places. Speaking of distractions, I also enjoyed the far too few, but unforgettable, co-working moments and office visits, as well as the usual craziness with Michelle. Finally, there is still one important name missing in this, already longer than expected, acknowledgment section. I am genuinely grateful to my girlfriend Laura, my partner on and off the dance floor, for the traveled journey and the road ahead. Thank you for sticking up with me, being able to cope with my very flexible and flue working hours, and enduring the technical conversations on multiple occasions. On a more practical note, I am also indebted for prepping my lunchbox in the mornings when I was still trying to get out of bed (which occurred quite often in the past two years) and, more recently, helping me out with constructing the RF-shielded box.

Antwerp, September 2019
Tom De Schepper

Samenvatting
– Summary in Dutch –

Tijdens de afgelopen decennia is er een enorme toename geweest in het gebruik en de beschikbaarheid van allerhande draadloze netwerken en toestellen. In 2017 waren er 18 miljard toestellen verbonden met het internet en men verwacht een verdere toename tot 28.5 miljard in 2022. Naast de meer traditionele toestellen zoals Personal Computers (PCs) en laptops, bestaat de grote meerderheid van de verbonden toestellen vandaag uit mobiele apparaten en toestellen gerelateerd aan het Internet der dingen (IoT). Voorbeelden hiervan zijn smartphones, tablets, en verschillende soorten sensoren. Verder blijft ook het aantal beschikbare communicatietechnologieën steeds groeien dankzij de ontwikkeling van nieuwe technologieën zoals 5G, IEEE 802.11ax, en IEEE 802.11ay. Het aantal en de variatie van toestellen en technologieën zal enkel toenemen, vooral wanneer steeds meer applicatiedomeinen, zoals slimme steden en industrie 4.0, het IoT paradigma uitrollen op grote schaal. In parallel, zijn de eisen en verwachtingen van gebruikers omtrent connectiviteit, bandbreedte, kwaliteit, en functionaliteit, nog nooit zo hoog geweest. Dit is in het bijzonder het geval voor moderne multimedia-applicaties, zoals Virtual Reality (VR), video op aanvraag (VOD), en gaming. Het is dus duidelijk dat deze evoluties hebben geleid tot een complexe en heterogene omgeving waar verschillende toestellen, applicaties, en technologieën naast elkaar bestaan op identieke of overlappende fysieke locaties. Hierdoor beconcurreren ze elkaar voor dezelfde draadloze hulpbronnen.

De bovenstaande situatie heeft geleid tot een toename van de managementlast en tot het volgende conflict: enerzijds, zijn er de strikte kwaliteitseisen die kenmerkend zijn voor moderne diensten en applicaties. Anderzijds, zijn er de unieke capaciteiten van communicatietechnologieën (b.v. de afstand en capaciteit) en toestellen (b.v. de ondersteunde technologieën). Helaas kan het bestaande netwerkbeheer typisch niet omgaan met de hele dynamische en heterogene omstandigheden van draadloze netwerken. Dit komt door de afwezigheid van coördinatie tussen verschillende communicatietechnologieën en toestellen. Ondanks dat moderne toestellen zijn uitgerust met verschillende netwerkinterfaces, opteren ze typisch voor het statisch selecteren van één van de beschikbare technologieën (b.v. Wi-Fi of LTE) of toegangspunten (b.v. access points (APs) of base stations) op basis van op voorhand gedefinieerde prioriteiten. Daarnaast, is de samenwerking tussen verschillende technologieën onmogelijk doordat een technologie kan gezien worden als een silo, geïsoleerd van andere technologieën. Dit is geen inherent probleem maar is een gevolg van de architectuur van de lagere lagen van het OSI-netwerkmodel. Nochtans is deze samenwerking nodig om functionaliteiten, zoals naadloze handovers tussen verschillende technologieën of de verdeling van de netwerkbelasting over verschillende connecties of componenten (load balancing), mogelijk te maken. Het gevolg is dat beslissingen worden overgelaten aan de applicaties, of nog erger, aan de gebruiker. Het is dus onmogelijk om automatisch te reageren op de onvermijdelijke dynamische onderbrekingen of veranderingen binnen de draadloze omgeving. Ten slotte, beconcurreren verschillende technologieën elkaar vaak binnen dezelfde frequenties. Dit is, bijvoorbeeld, het geval voor Wi-Fi, ZigBee, en Bluetooth binnen de 2.4 GHz frequentie. In het algemeen, is het duidelijk dat er nood is aan intelligent beheer over de verschillende technologieën heen en het meer efficiënt gebruiken van de draadloze hulpbronnen.

Er bestaan reeds een aantal controle- en beheersoplossingen hiervoor, maar deze zijn vaak afhankelijk van een bepaalde technologie en richten zich op een specifiek soort netwerk domein of use case. Hoewel sommige oplossingen in staat zijn om te coördineren over heel het netwerk, bieden ze meestal slechts controle aan op het niveau van datastromen. De meest noemenswaardige oplossingen zijn MPTCP, de IEEE 1905.1 standaard, LTE-LWA, en op Software-Defined Networking (SDN)-gebaseerde oplossingen. Deze oplossingen bieden typisch enkel beheersfunctionaliteiten (b.v. handovers) aan en bevatten geen intelligentie of algoritmen die effectief in staat zijn om het netwerk te gaan optimaliseren, zoals het bepalen van het meeste geschikte pad voor elke datastroom. Bestaand werk omtrent het verdelen van de netwerkbelasting binnen heterogene netwerken focust meestal op de ontwikkeling van theoretische modellen en houdt geen rekening met de bijzondere aard van draadloze netwerken. In tegenstelling tot de bestaande oplossingen, proberen we in dit proefschrift de opgenoemde uitdagingen op te lossen in een transparante en fundamentele manier. De belangrijkste leidraad hierbij is dat een gebruiker zich enkel bewust zou mogen zijn van het feit dat zijn toestel verbonden is met het internet, terwijl het netwerk alle onderliggende beslissingen neemt en de benodigde voorzieningen beschikbaar maakt. Als zodanig, wordt er een betere Quality of Service (QoS) en gebruikservaring aangeboden.

Een eerste contributie is de introductie van het ORCHESTRA-raamwerk dat naadloos beheer over verschillende technologieën heen mogelijk maakt. De oplossing bestaat uit twee componenten: een virtuele MAC (VMAC) laag en een gecentraliseerde controller. De VMAC zorgt voor een enkel verbindingspunt voor de hogere lagen door de onderliggende netwerktechnologieën op een transparante manier samen te voegen. Dit maakt handovers tussen de verschillende technologieën, het verdelen van de netwerkbelasting, en de duplicatie van datastromen mogelijk op het niveau van individuele datapakketten. Hiervoor maken we gebruik van regels die worden afgetoetst aan elk pakket. Daarnaast zorgt de gecentraliseerde controller voor een globaal overzicht van het netwerk door de ontvangst van gedetailleerde monitoringsinformatie van elke VMAC. Omgekeerd, kan de controller de netwerkconfiguratie aanpassen door instructies te geven aan individuele VMACs, bijvoorbeeld om specifieke regels aan te passen. Om een graduele uitrol mogelijk te maken, ondersteunt de ORCHESTRA-omgeving communicatie met reeds bestaande netwerkcontrollers en gaat het transparant om met legacy toestellen. We stellen een alledaags prototype voor dat in staat is om de verschillende voorgestelde functionaliteiten te demonstreren over verschillende communicatietechnologieën zoals Ethernet, Wi-Fi, en LTE. Uit de evaluatie blijkt dat onze oplossing beter presenteert dan MPTCP, de standaardoplossing gebruikt door de industrie. Dit terwijl ook UDP-dataverkeer mogelijk is.

Bovenop de geïntroduceerde ORCHESTRA-oplossing, is er nood aan intelligentie die in staat is om de functionaliteiten van het raamwerk te gebruiken om het netwerk optimaal te gaan configureren. Daarom stellen we een aantal algoritmen voor die trachten de toegelaten hoeveelheid verkeer (throughput) binnen een netwerk te verhogen. We maken in het bijzonder gebruik van methodes die steunen op lineair programmeren om op een wiskundige manier het probleem te formuleren. Hierbij is het essentieel dat we de specifieke aard van draadloze netwerken (b.v. de impact op de totale bandbreedte van verschillende toestellen die elkaar beconcurreren) in rekening brengen. Een eerste algoritme concentreert zich op het berekenen van de beste paden voor datastromen over verschillende Ethernet en Wi-Fi verbindingen. Uitgebreide NS-3 simulaties en een kleinschalig prototype tonen aan dat gemiddeld een verbetering van 20 % mogelijk is, in vergelijk met een statische baseline. Nadien, breiden we dit algoritme uit om rekening te houden met verschillende toegangspunten en de mobiliteit van de gebruikers. We tonen aan dat het optimalisatieprobleem om verschillende gebruikers te verdelen over verschillende toegangspunten en het plannen van datastromen over verschillende verbindingen en technologieën, exponentieel schaalt. Daarom stellen we twee heuristieken voor die het probleem opsplitsen en de deelproblemen sequentieel oplossen. We tonen aan dat de greedy heuristiek in staat is om meer dan dubbel zoveel dataverkeer mogelijk te maken binnen het netwerk en ook schaalbaar is tot netwerken met 10000 toestellen.

Ten slotte, onderzoeken we of het mogelijk is om dataverkeerspatronen te detecteren in het radiospectrum. Deze informatie kan gebruikt worden door beheersalgoritmen om het netwerk verder te optimaliseren, in het bijzonder in situaties met verschillende overlappende netwerken. Deze methode staat haaks op traditionele methodes zoals Deep Packet Inspection (DPI) met een meer intrusief gedrag. We ontwerpen een Deep Learning architectuur die gebruikt wordt om drie verschillende classificatiemodellen te bouwen. Deze modellen zijn, respectievelijk, in staat om TCP- en UDP-datastromen, datastromen met verschillende patronen, en datastromen met verschillende groottes (rates) te onderscheiden. Als invoer voor deze modellen vergelijken we het gebruik van afbeeldingen die het spectrum voorstellen doorheen de tijd of als spectrogrammen. Daarnaast stellen we ook een nieuwe datarandomisatiemethode voor die het mogelijk maakt om op grote schaal synthetische data te genereren door het combineren van twee recente simulatoren. Een evaluatie met een test dataset, bestaande uit synthetisch gegenereerde data, toont aan dat alle modellen een accuraatheid van minstens 96 % behalen. Verder is het mogelijk om binnen een realistische omgeving nog verschillende groottes van datastromen (rates) te herkennen met een accuraatheid van ongeveer 87 %.

Samengevat, stelt dit proefschrift meerdere verbeteringen voor binnen de context van netwerkbeheer over verschillende communicatietechnologieën heen. Er zijn verschillende contributies: als eerste stellen we de ORCHESTRA-omgeving voor die verschillende functionaliteiten biedt, zoals naadloze handovers tussen verschillende technologieën of de verdeling van de netwerkbelasting op pakketniveau. Verder ontwikkelen we meerdere algoritmes die bovenop ORCHESTRA gebruikt kunnen worden om het netwerk te optimaliseren en meer verkeer binnen een netwerk mogelijk te maken. Ten slotte, introduceren we verschillende modellen om specifieke informatie van datastromen te detecteren in het radiospectrum.

Summary

Over the last decades, we have witnessed a tremendous increase in the utilization and availability of wireless networks and devices. The number of connected devices has reached 18 billion in 2017 and it is predicted that it will further grow to 28.5 billion in 2022. Besides the more traditional devices like Personal Computers (PCs) and laptops, the vast majority of connected devices these days are mobile and Internet of Things (IoT) devices, such as smartphones, tablets, and all kinds of sensors. Similarly, the number of available communication technologies is constantly growing as well due to the releases of new technologies such as 5G, IEEE 802.11ax, and IEEE 802.11ay. The amount and variety of devices and technologies will only grow, especially when more and more application domains, such as smart cities and Industry 4.0, adopt the IoT paradigm on a larger scale. In parallel, the demands and expectations of users, in terms of connectivity, bandwidth, quality, and services, have never been higher. This is particularly the case for modern multimedia applications, like Virtual Reality (VR), Video-On-Demand (VOD), and gaming. It is clear that these evolutions have led to a complex and heterogeneous situation where different devices, applications, and technologies consist next to each other at identical or overlapping physical locations, often competing for the same wireless resources.

The aforementioned situation has increased the management burden and raised the following conflict: at one hand the stringent quality requirements that are typical for modern services and applications. On the other hand, the unique capabilities of communication technologies (e.g., range and capacity) and devices (e.g., supported technologies). Unfortunately is existing network management typically unable to cope with the very dynamic and heterogeneous conditions of wireless networks, as no coordination exists across different communication technologies and devices. Despite being equipped with multiple network interfaces, modern devices tend to statically select one of the available technologies (e.g., Wi-Fi or LTE) or connection points (e.g., access points (APs) or base stations) based on predefined priorities. Similarly, cooperation across different technologies to enable features like seamless handovers or multi-path load balancing is not possible as each technology can be seen as a silo, isolated from the others. The latter is not an inherent problem but is due to the design of the lower layers of the OSI network stack. As such, decisions are left to the applications or, even worse, the user and it is impossible to automatically react to the inevitable dynamic disruptions or changes in the wireless context. Furthermore, different technologies tend to compete against each other in the same frequency bands. This is, among others,

the case for Wi-Fi, ZigBee, and Bluetooth in the 2.4 GHz band. Overall, it is clear that there is a need for intelligent inter-technology management and more efficient use of wireless resources.

A number of control and management approaches already exist, however, they are typically technology dependent and target a specific network domain or use case. While some solutions do offer network-wide coordination, they typically only offer flow-level control over the network. The most notable solutions are MPTCP, the IEEE 1905.1 standard, LTE-LWA, and Software-Defined Networking (SDN)-based solutions. Furthermore, these solutions typically only enable management features (e.g., handovers) but do not contain the intelligence or algorithms to actually optimize the network, such as selecting the most suitable path(s) per traffic flow. Existing work on load balancing in heterogeneous networks focuses mostly on the development of theoretical models and neglects the specific nature of wireless networks. In contrast, in this dissertation, we try to tackle the aforementioned challenges in a more transparent and fundamental manner. The main idea is that a user is only aware of the fact that its device is connected to the Internet, while the network takes care of all the underlying decision-making and the provisioning of resources. As such, an improved Quality of Service (QoS) and user experience is offered.

A first contribution is the introduction of the ORCHESTRA framework for seamless inter-technology management. The framework consists of two components: a Virtual MAC (VMAC) layer and a centralized controller. The VMAC offers a single connection point to the upper layers, while transparently bonding over the underlying network technologies. As such, it introduces packet-level inter-technology handovers, load balancing, and duplication across the different underlying links by using packet matching rules. Furthermore, the centralized controller maintains a global network overview by receiving detailed monitoring information from each VMAC and can in return enforce instructions (i.e., propagate rule changes). In order to support a gradual network-wide roll-out, the framework supports communication with existing network controllers and operates transparently towards legacy devices. We present a real-life prototype that is capable of demonstrating the different proposed features across different communication technologies like Ethernet, Wi-Fi, and LTE. A performance evaluation shows that the ORCHESTRA framework outperforms the default industry solution MPTCP, while also supporting UDP traffic.

On top of the introduced ORCHESTRA framework, intelligence is needed that can utilize the features of the framework to actually optimize the network. To this extent, we present a number of algorithms that aim to increase the network-wide throughput based purely on real-time monitoring information. In particular, we make use of linear programming approaches to construct mathematical problem formulations. Key in this is that the specific nature of wireless networks (e.g., the impact of competing stations on the maximum capacity of the network) is considered. A first algorithm focuses on calculating the best paths for traffic flows across Ethernet and Wi-Fi links. Extensive NS-3 simulations and a small-scale prototype show that improvements of on average 20 % can be made, in comparison to a static

baseline. Afterwards, we extend this algorithm in order to include multiple connection points and client mobility. We show that the problem of optimally load balancing stations across different infrastructure devices and scheduling flows across different connections and technologies scales exponentially. Consequently, we present two heuristic approaches that sequentially solve these two problems. We show that the greedy heuristic can more than double the network-wide throughput and can scale up to scenarios of 10000 devices.

Finally, we explore the option of detecting traffic patterns in the wireless spectrum. This information can be used by management algorithms to further optimize the network, especially in areas with multiple overlapping networks. This strongly contrasts with more-intrusive methods like Deep Packet Inspection (DPI). In particular, we design a Deep Learning architecture to construct three different classification models that differentiate between TCP and UDP traffic, burst traffic with different duty cycles, and traffic with different transmission rates. As input to these models, we explore the use of images that represent the spectrum in time and time-frequency domain. Furthermore, we present a novel data randomization approach to generate large amounts of synthetic data by combining two state-of-the-art simulators. Validation with a synthetic test dataset shows that all models achieve an accuracy of above 96 %. In a real-life setting, it is still possible to recognize different rates with an accuracy of around 87 %.

To summarize, this dissertation provides improvements in the area of inter-technology network management. Different contributions are made: first, the ORCHESTRA framework enables different multi-technology management features such as seamless handovers or load balancing. Next, we present different algorithms that can be deployed on top of this framework to optimize the configuration of the network and to increase the network-wide throughput. Finally, we present different models to detect traffic information in the wireless spectrum.

Contents

List of Figures

List of Tables

Acronyms

Symbols

2G	Second Generation
3G	Third Generation
3GPP	Third Generation Partnership Project
4G	Fourth Generation
5G	Fifth Generation

A

ACK	Acknowledgement
AP	Access Point
API	Application Programming Interface
AR	Augmented Reality
ARP	Address Resolution Protocol
AWGN	Additive White Gaussian Noise

B

BGP	Border Gateway Protocol
BLE	Bluetooth Low Energy
BSS	Basic Service Set

C

CBRS	Citizens Broadband Radio Service
CNN	Convolutional Neural Network
CR	Cognitive Radio

CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
D	
DHCP	Dynamic Host Configuration Protocol
DL	Deep Learning
DNN	Deep Neural Network
DPI	Deep Packet Inspection
DSA	Dynamic Spectrum Access
DSL	Digital Subscriber Line
E	
eNB	Evolved Node B
EPC	Evolved Packet Core
F	
FBSF	Flow-Based Scheduling Formulation
FDD	Frequency Division Duplex
FFT	Fast Fourier transform
FTP	File Transfer Protocol
G	
GAN	Generative Adversarial Network
GBSF	Group-Based Scheduling Formulation
GEO	Geosynchronous
GPRS	General Packet Radio Service
GPU	Graphics Processing Unit
GSM	Global System for Mobile communications
GTP	GPRS Tunneling Protocol
H	
HetNet	Heterogeneous Network

HP HomePlug

I

ILP Integer Linear Programming

IoT Internet of Things

IP Internet Protocol

IQ In-phase and Quadrature

ISP Internet Service Provider

L

LAN Local Area Network

LBO Local Breakout

LBT Listen-Before-Talk

LCG Link Collision Group

LEO Low Earth Orbit

LLDP Link Layer Discovery Protocol

LowRTT Lowest Round Trip Time First

LSTM Long Short-Term Memory

LTE Long-Term Evolution

LTE-LAA LTE License Assisted Access

LTE-U LTE-Unlicensed

LTE-V Long-Term Evolution-Vehicular

LVAP Light Virtual AP

LWA LTE-WLAN Aggregation

LWAAP LTE-WLAN Aggregation Adaptation Protocol

M

M2M Machine-to-Machine

MAC Media Access Control

MCS Modulation and Coding Scheme

MEC Multi-access Edge Computing

MF-TDMA Multi-Frequency Time-Division Multiple Access

MIH Media Independent Handover

MIHF Media Independent Handover Function

MILP Mixed Integer Linear Programming

MIMO Multiple-input and Multiple-output

MINLP Mixed Integer Non Linear Programming

ML Machine Learning

MMS Multimedia Messaging Service

mmWave Millimeter-Wave

MoCA Multimedia over Coax

MPTCP Multipath Transmission Control Protocol

N

NAS Network-attached Storage

NAT Network Address Translation

NETCONF Network Configuration Protocol

NFC Near Field Communication

NFV Network Function Virtualization

NN Neural Network

NR New Radio

O

OFDM Orthogonal Frequency Division Multiplexing

OS Operating System

OSI Open Systems Interconnection

OVS Open vSwitch

P

P2P Peer-to-Peer

PAN Personal Area Network

PC Personal Computer

PDA Personal Digital Assistant

PDCP Packet Data Convergence Protocol

PLC Power line communication

PNG Portable Network Graphics

Q

QoS Quality of Service

R

RAN Radio Access Network

ReLU Rectified Linear Unit

RF Radio Frequency

RSSI Received Signal Strength Indicator

RTT Round Trip Time

S

SAP Service Access Point

SCG Station Collision Group

SD-WAN Software-Defined Wide Area Networks

SDN Software-Defined Networking

SDR Software-Defined Radio

SIP Session Initiation Protocol

SL Supervised Learning

SMS Short Message Service

SNR Signal-to-noise Ratio

STFT Short Time Fourier Transform

T

TCP	Transmission Control Protocol
TDD	Time Division Duplex
TDMA	Time-Division Multiple Access
U	
UDP	User Datagram Protocol
UE	User Equipment
UMTS	Universal Mobile Telecommunications Service
UTP	Unshielded Twisted Pair
V	
VAN	Virtualized Access Network
VANET	Vehicular AdHoc Network
VAP	Virtual AP
VMAC	Virtual MAC
VNF	Virtualized Network Function
VOD	Video on Demand
VoIP	Voice over IP
VoLTE	Voice over LTE
VR	Virtual Reality
W	
WAN	Wide Area Network
Wi-Fi	IEEE 802.11
WiMAX	IEEE 802.16 Worldwide Interoperability for Microwave Access
WWW	World Wide Web

1 Introduction

“The beginning is the most important part of the work”

– Plato (427 - 347 B.C.)

1.1 Context

While the year 1969 is most famous for the first moon landing, it also marked the beginning of the Internet. In October 1969, for the first time in history, two remote computers were connected with each other over the so-called ARPANET [1]. This was the first network where data was divided into various (smaller) packets that could be routed differently across the links of the network. Such a type of network is known as a packet-switching network [1, 2]. The number of connected computers grew steadily from 4 by the end of 1969 to 213 in 1981. In order to provide connectivity across different physical sites and networks, such as ground and satellite networks, the Transmission Control Protocol (TCP) / Internet Protocol (IP) stack and protocols were introduced [1, 2]. Thus, allowing for an end-to-end connection, abstracting away the internal structure of the network and its building blocks. Although the ARPANET was decommissioned in 1990, it was the foundation of the Internet as we know it and key principles, such as robustness and encapsulation, have been mainstream ever since.

Over time, the number of devices connected to the Internet continued to grow: from around 100000 devices at the beginning of the nineties when the World Wide Web (WWW) was born, up to 500 million in 2003 [1,3]. The number of connected devices surpassed the number of human beings in 2008 and nearly doubled by 2012, reaching 12.5 billion [3]. Five years later, in 2017, there were 18 billion

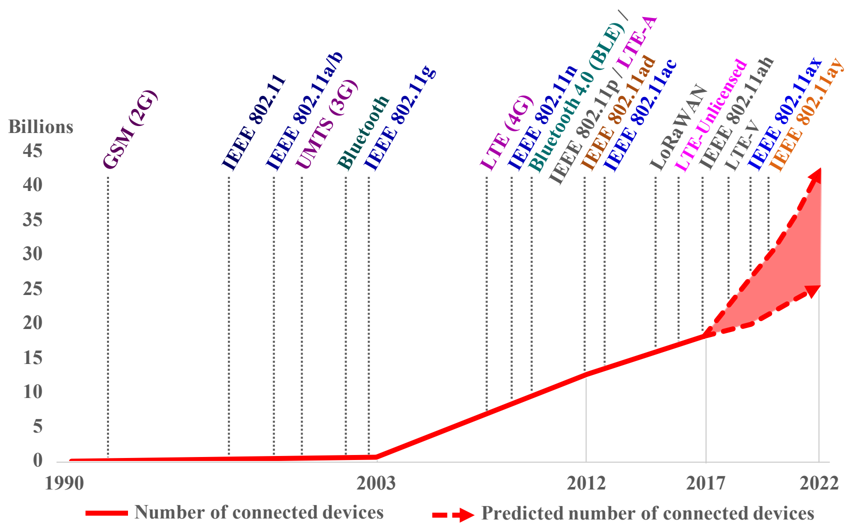


Figure 1.1: A timeline showing the exponential growth of the number of connected devices and the consecutive release dates of a selection of important wireless communication technologies [7, 10–14]. Releases of the same technology are highlighted in the same color, with the earliest release highlighted by the darkest color tone, and the most recent one in the lightest tone. Technologies without a follow-up release listed are colored in grey.

devices connected to the Internet and it is expected that this number will grow further and reach between 25.4 and 42.6 billion by 2022 [4–6]. Cisco keeps it at 28.5 billion by 2022, or 3.6 times the world's population [4]. This great expansion over time, as depicted in Figure 1.1, is founded on two major technology shifts: mobile devices and the Internet of Things (IoT) [1, 2]. While the first decades mostly traditional devices like Personal Computers (PCs) were connected to the Internet, this changed with the introduction of devices such as cellphones, Personal Digital Assistants (PDAs), and laptops in the second half of the nineties. The real boost occurred with the massive worldwide adoption of smartphones and tablets, which allowed for relatively cheap and easy Internet access [3, 7]. Nowadays, mobile devices are everywhere and account for a large portion of all consumer devices (8.6 billion on a total of 18 billion in 2017) [4]. IoT is a second major paradigm shift where all kinds of everyday devices are being connected to the Internet [8]. While the idea was launched earlier, the adoption and roll-out only started the last decade, for instance, with the releases of smart televisions and home automation [3, 8]. The amount and variety across IoT devices will only grow, when more and more application domains, such as smart cities and industry 4.0, adopt the new paradigm on a larger scale [4, 9].

Similar to the connected devices, the communication technologies that can be used to connect to the Internet have also evolved drastically. This is illustrated in Figure 1.1 that lists a selection of important (wireless) communication technology releases throughout the last 20 years. Going back to the beginning years of the Internet, connecting a machine to the ARPANET was only possible using a wired connection. While wired technologies (i.e., Ethernet) can still be found in, for instance, Local Area Networks (LANs), wireless technologies have gained massive popularity, with IEEE 802.11 (Wi-Fi) and cellular being the two most dominant technologies [10–12]. The most available technology, up to now, has been Wi-Fi because of its ease of use, relatively low cost, flexibility, and mobility [10, 11]. Since the first release, in 1997, a multitude of different Wi-Fi standards have been released, as depicted in Figure 1.1 [7]. Consecutive releases of Wi-Fi, highlighted in different tones of blue, have mainly focused on increasing the capacity (in terms of bandwidth), Quality of Service (QoS), and the number of supported users in order to meet growing customer demands. Recent releases have diversified from the mainstream Wi-Fi in order to address specific needs like very high throughput (i.e., IEEE 802.11ad) and low power (i.e., IEEE 802.11ah). In the coming years, a number of follow-up releases are expected to further increase the QoS and user experience, while more efficiently using the wireless spectrum (e.g., IEEE 802.11ax, IEEE 802.11ay, ...) [10, 11]. In parallel, the original Second Generation (2G) standard (including Global System for Mobile communications (GSM) and General Packet Radio Service (GPRS) technologies) that allowed for digital services like Short Message Service (SMS) and Multimedia Messaging Service (MMS), has evolved into the Third Generation (3G) (i.e., Universal Mobile Telecommunications Service (UMTS)) and Fourth Generation (4G) (i.e., Long-Term Evolution (LTE)) standards [12, 15]. All of them are highlighted in different shades of purple and pink. These newer standards introduced worldwide mobile Internet access via cellular networks, offered by all telecommunication operators. The highly anticipated Fifth Generation (5G) technology will be the next step in this evolution, offering, among others, massive bandwidths and high-speed mobile connections [12]. As such, it will drastically change our ways of communication and allow for new interactions and applications, both between humans mutually, between humans and machines, and between machines mutually. Furthermore, the IoT paradigm has also introduced a number of specific wireless communication technologies. For instance, LoRaWAN and Sigfox for long-range Machine-to-Machine (M2M) communication, Bluetooth Low Energy (BLE) and ZigBee for short-range communication (e.g., for wearables or home automation), and IEEE 802.11p and Long-Term Evolution-Vehicular (LTE-V) for vehicular communications [13, 14].

The increased number and variety of connected devices and communication technologies have enabled the development of a large multitude of different applications. After the launch of the ARPANET, applications such as File Transfer Protocol (FTP) and email were first developed [1,2]. This was followed by web browsing and Peer-to-Peer (P2P) services like Napster in the nineties. When bandwidth capacities and connection speeds increased with the release of new technologies in the 2000s, multimedia services started to appear. These multimedia services have dominated the Internet traffic landscape ever since, in particular since the rise of mobile devices that are equipped with all kinds of actuators, sensors, and features. Currently, 75 % of all Internet traffic is video traffic and this is expected to rise to 82 % by 2022 [4]. Multimedia applications like Virtual Reality (VR), Augmented Reality (AR), Video on Demand (VOD), and gaming will only increase in popularity the next years [4]. With this boost of popularity, the expectations and QoS demands of end-users and companies also raised [16]. In the case of multimedia services, key aspects in order to have a smooth video experience are high throughput and low latency. The IoT paradigm has also boosted the Internet traffic by introducing new applications relying particularly on M2M communications [13]. Examples of such applications are control, either manually or automatically, of the heating, air-conditioning, lights, and other home appliances. Similarly, based on sensor data, production lines, robots, and other machines can be controlled on a fine-grained level in an Industry 4.0 setting. Low power and low latency communications are key for these types of applications [17]. Furthermore, a large number of other applications exist, such as vehicular applications (e.g., platooning, infotainment, and safety messaging) that demand reliable real-time communications [14]. The amount of and diversity among applications and their demands will only grow further in the future [4, 10, 16].

The previous paragraphs have clearly illustrated how the Internet has drastically evolved throughout the years, with a tremendous increase in the utilization and availability of (mostly wireless) communication technologies, connected devices, and applications. This has led to the complex and heterogeneous situation of today where different devices, applications, and technologies consist next to each other at identical or overlapping physical locations, often competing for the same resources such as bandwidth and airtime [18, 19]. Furthermore, most modern devices are equipped with multiple communication technologies that can be employed to connect to the Internet or to communicate with other devices. This is, for instance, the case for smartphones that are equipped with, among others, multiple Wi-Fi standards, LTE, BLE, and Near Field Communication (NFC). Each of these devices, applications, and communication technologies has unique capabilities (e.g., bandwidth and range) and requirements (e.g., latency or power consumption) that should be respected [10, 17, 20]. Not respecting these capabilities and requirements can lead to significant QoS disruptions, which in turn can cascade into a dramatic impact on user experience and application behavior. As the diversity and number of devices and technologies, and the stringent requirement of applications will only grow in the future, the complexity will consequently increase as well. This scenario is depicted in Figure 1.2 where a large number of networks coexist next to each other and are occupied with a large variety of devices. At the left top side of the figure we see a LAN environment, consisting of multiple Access Points (APs) offering a number of different wireless technologies (e.g., directional 60 GHz communication, a sub-1 GHz low power technology, and more standard 2.4 GHz and 5 GHz Wi-Fi). Similarly, at the top right side of Figure 1.2, we show an Industry 4.0 use case where a combination of IoT technologies and Wi-Fi is used. Furthermore, we see a number of Vehicular AdHoc Networks (VANETs) (using IEEE 802.11p or LTE-V) that provide both connectiv-

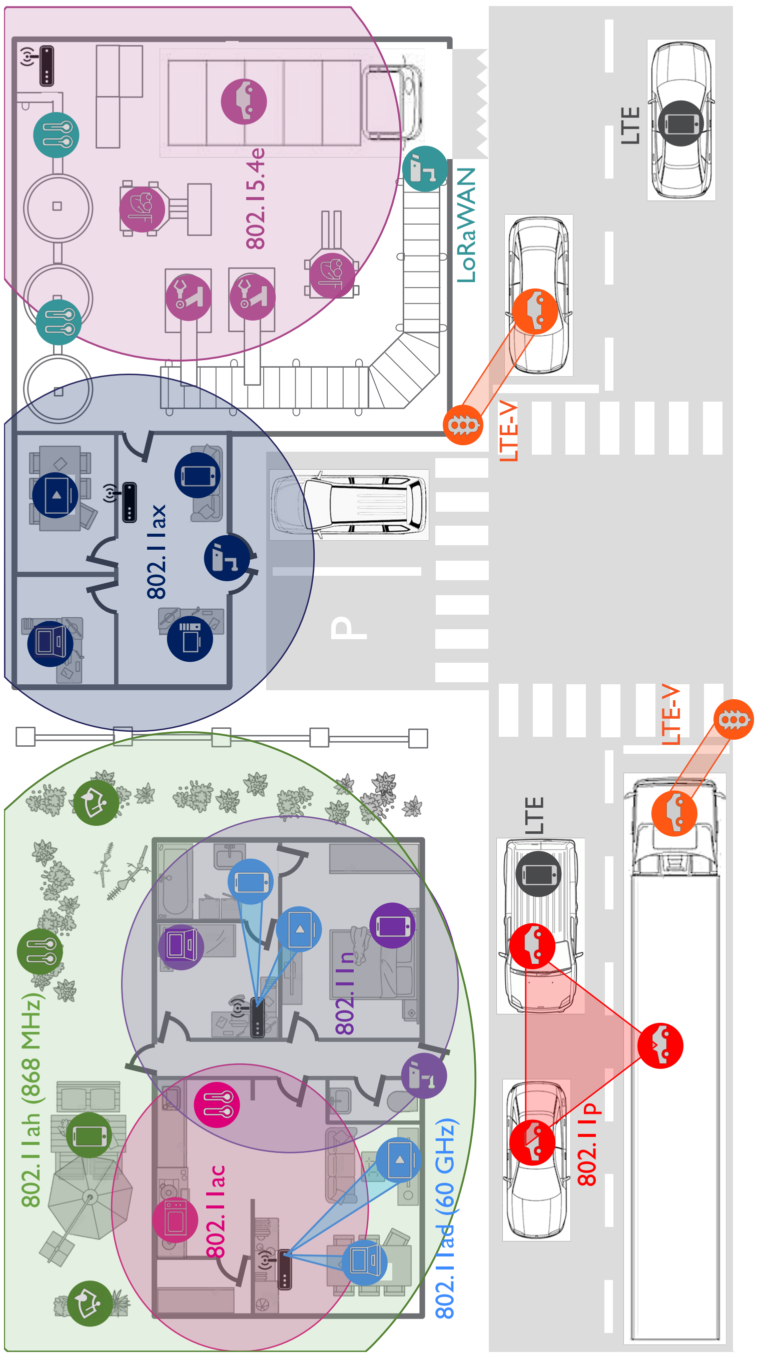


Figure 1.2: Example of a near-future environment, consisting of a large variety of different heterogeneous networks, application domains, technologies, and devices.

ity between different vehicles mutually and between vehicles and road-side units while requiring reliable real-time communication.

1.2 Problem Statement

As presented in the previous section, the wireless networks of today and tomorrow are continuously becoming more heterogeneous and complex. This has increased the management burden and raised the following conflict: at one hand the stringent quality requirements that are typical for modern services and applications. On the other hand, the unique capabilities of communication technologies (e.g., range and capacity) and devices (e.g., supported technologies). Furthermore, these networks are typically being managed statically, delegating decisions to the application layer, or even worse, to the user. This makes it impossible to automatically react in a timely fashion to disruptions that cause QoS degradations. As both modern connected devices (like consumer, infrastructure, or backhauling devices) and wireless networks are equipped with multiple communication technologies, allowing devices to simultaneously use different technologies or to switch in real-time between them, would lead to network optimizations. Such optimizations, like multipath routing, load balancing, and dynamic path reconfiguration, can increase the overall network performance, QoS, and end-user experience [21]. In this dissertation, we focus on providing both the technical innovations to enable such optimizations and the network management algorithms to achieve them. More specifically, the following problems are identified and tackled in this thesis:

1. **Communication technologies operate fully independently of each other.** As introduced in the previous section and indicated in Figure 1.1, the set of available communication technologies is expanding rapidly. The Open Systems Interconnection (OSI) model is the standard model for end-to-end network communication and partitions a communication system into 7 different abstraction layers, each with a specific purpose and functionality, in order to achieve interoperability of different systems, standards, and networks [7, 22]. However, each communication technology (e.g., Wi-Fi or LTE) defines its own lower layers of this network stack (in particular the Media Access Control (MAC) and physical layers) and gets assigned individual network addresses. Each of these technologies operates completely independently, isolated from each other, making cooperation between them infeasible [22–24]. This isolation is not an inherent problem but is due to the design of the lower layers of the OSI network stack. This despite the fact that cooperation between technologies is needed to enable optimization features such as inter-technology handovers (also known as vertical handovers) or load balancing. These inter-technology handovers are key to enable seamless mobility of devices and traffic offloading across different networks or links [25, 26]. Besides the independent operation of technologies, there are also multiple management actors present across different use cases (e.g.,

homeowners, enterprises, or telecommunication operators), each with their expectations and priorities. Furthermore, limited by the amount of available spectrum, more and more technologies are operating within each other's frequency bands, which leads to significant QoS degradations and performance loss [18, 19, 24]. This is, for instance, the case for LTE and Wi-Fi in the 5 GHz frequency band and Wi-Fi, ZigBee, and Bluetooth in the 2.4 GHz frequency band [19, 27, 28]. In order to fully utilize the available resources (e.g., bandwidth) of, in particular, wireless technologies, there is the need for intelligent and fine-grained control over both devices and traffic streams.

2. **Autonomous and real-time coordination, across the different devices in a network, is missing.** As mentioned above, (wireless) networks are currently typically being managed in a static manner. In general, each device decides for itself to which infrastructure device (e.g., AP or base station) it will connect [29]. This leads to sub-optimal behavior and unfairly balanced network traffic [21, 30]. Furthermore, the decision to activate and use a specific technology, or to perform a handover between different technologies, is left to the Operating System (OS), application (e.g., by binding a socket on a specific interface), or user (e.g., by means of a button). This static behavior makes it impossible to react to dynamic changes to the network environments. Examples of such dynamic events are, among others, link failure, congestion, and arrivals or departures of devices and traffic flows. Autonomous traffic rerouting mechanisms are needed to cope with such events and lift responsibility away from the user and applications [31]. Moreover, coordinated real-time intelligence can help in fairly distributing the load across different technologies and infrastructure devices [21, 31]. The efficiency of such a dynamic approach will only increase as more and more technologies, such as 60 GHz and sub-1 GHz communications, are introduced to the environment of wireless networks. However, it is key that these intelligent mechanisms take into account the specific nature of wireless technologies (e.g., the shared bandwidth amongst different devices) [26,32].

3. **Tailored management systems, that account for mobility and the ever-growing set of devices, are lacking.** The growth of the networks of today and tomorrow is mainly funded by the shift towards mobile and IoT devices and networks, as discussed in Section 1.1. Both paradigms introduce additional challenges to the network management systems [3,10,17,26,29]. This complements the need for seamless mobility on a per-device level, as discussed in the first problem listed in this section. First of all, the heterogeneity of the network further increases because of the strong differences, across the different devices: the speed differs (e.g., a connected car compared to a cleaning robot), the movement patterns are unique (an autonomous drone compared to a laptop), and there are still devices that are not mobile at all (e.g., a smart television or production line machines). Second, because of the mobility, the load across different infrastructure devices can vary heavily over time. When a single device moves from one area to another (for

instance, different rooms or floors within a building), it is possible that a connection is established to an already saturated infrastructure device or network. Consequently, the performance and experience of both the moving device and the already present devices can be highly impacted. For larger groups of mobile devices, the problem will only increase. Note that these two raised challenges clearly further complicate the already existing management problem related to the different requirements and capabilities of devices and technologies, as described in the previous paragraph. Additionally, the ever-growing amount of devices is even more cumbersome, as management systems need to support this growth and scale accordingly. In particular, the previously introduced real-time coordination still needs to be feasible within a network of, for instance, hundreds, thousands, or even more devices. Examples of such networks are, among others, large office or conference networks, Industry 4.0 deployments within a harbor, or a vehicular network on a highway [4, 8, 9]. Network operators that offer multiple (wireless) technologies can, among others, benefit from these tailored management systems. Especially, since we notice that telecommunication operators increasingly often provide, next to the traditional wired en cellular connections, also Wi-Fi technologies (e.g., hotspots) and IoT technologies.

4. **The coexistence of neighboring technologies and networks is heavily being pressured.** In Figure 1.2 we have shown how different types of networks are coexisting next to each other. With the ever-rising number of devices, technologies, and applications, this coexistence is being pressured [18, 19]. Within the problem context of the independent operation of technologies, we already highlighted the potential QoS degradations and performance loss when technologies are competing with each other in the same frequency bands [10, 18, 24]. For instance, the direct use of LTE in the unlicensed spectrum (i.e., LTE-U) could lead to severe performance degradation (up to more than 90 %) of coexisting Wi-Fi systems, especially in the 5 GHz band [19, 33]. However, even when inter-technology management mechanisms are in place, the question remains how to cope with neighboring networks that are not under your control. Examples of such uncontrollable networks are neighboring LANs in a residential area or apartment building or, from an operator point of view, the network of a competitor. The latter is, for instance, the case when different operators deploy LTE technologies in the 5 GHz frequency band or when different actors operate in the 3.5 GHz band (i.e., Citizens Broadband Radio Service (CBRS)). Furthermore, for instance for Wi-Fi, it is possible to divide different APs across distinct channels, but there are only a limited amount of different channels available. This means that for crowded areas (e.g., an apartment building) it is impossible to provide a single channel per infrastructure device. Note that while operating on separate channels, two APs can still interfere with each other, for instance, due to inappropriate transmit power levels [34]. Additionally, technologies like Wi-Fi can also suffer from external interference sources

such as microwaves, cordless phones, or game controllers [35]. As such it is clear that there is a need for the efficient and intelligent use of spectral resources [10,36]. In order to have such adaptive management systems, it is crucial that these systems have access to accurate and real-time information of the state of the wireless spectrum.

1.3 Hypothesis

In the previous section, we have identified four distinct problems within the scope of the heterogeneous (wireless) networks of today and tomorrow. In order to address these issues and to provide the next generation of Internet communication, we formulate the following hypothesis:

Intelligent and dynamic inter-technology network management is needed to support the ever-evolving heterogeneous wireless networks.

We have mentioned previously that there is **no coordination and cooperation among the different adjacent communication technologies** on modern communication devices due to, among others, the design of the lower layers of the OSI network stack. As a result, the scarce inter-technology management takes place at the higher layers of the stack, following either a device driven or infrastructure driven approach. In the first case, the management is fully distributed and it is up to the devices (i.e. the application, operating system or user) to decide which technology will be used and, for instance, to which infrastructure device a connection will be established [25, 37]. In the second approach, the management control resides in the network infrastructure or the cloud and enforces certain policies or actions on the devices without user or application intervention. This is, for instance, the case with band steering to push certain devices to a specific technology or the use of tunnels to hide the underlying technology. The traditional manner to cope with the ever-growing stringent requirements of users and modern (multimedia) services has always been to boost the capacity by expanding the set of available frequencies and channels, increasing the channel widths, or introducing additional antennas [7,10]. This despite the significant changes required to the available hardware. However, as mentioned previously, the increased amount of technologies has lead to radio spectrum scarcity since both licensed and unlicensed spectrum bands are getting extremely crowded [18, 19, 24]. Furthermore, the over-provisioning of, among others, connection points is typically not beneficial in a wireless context. Existing approaches fail to deliver the required fine-grained seamless management and are often only applicable to certain technologies or networks. While they are also not fit to cope with the growing number of devices and neighboring technologies, nor provide the necessary flexibility.

In stark contrast, we envision an approach that solves the problem of managing different communication technologies in a more fundamental manner that is also adaptable and robust towards future evolutions within the area of, in particular, wireless communications. We believe that **it should not be up to the user or application to make network connectivity decisions**, such as which technology to use or to which infrastructure device to connect to [38, 39]. Instead, a single connection should be offered to the higher layers that combines the underlying technologies in a transparent manner. This would allow for seamless management, such as handovers, as the underlying technologies are completely abstracted away. Essentially, the encapsulation and abstraction principles introduced by the ARPANET and the OSI stack are extended and evolved to the multi-technology nature of current and future networks.

In addition, **a centralized entity offering advanced intelligence should be introduced in the network** [26]. This entity can dynamically enforce certain policies and instructions (e.g., the roll-out of an inter-technology handover) on the different heterogeneous devices, independent of the type of these devices (e.g., consumer, infrastructure, or backhaul). Such a centralized entity is feasible as more and more technologies are enrolled next to each other, under control by the same operator. It allows for the introduction of intelligent routing of traffic flows and load balancing of devices to the network in order to optimize network-wide performance. The envisioned routing approach will redistribute traffic flows across the existing connections in the entire network when needed to meet the traffic demands and respond to dynamic events such as link failure or increasing traffic volumes. Furthermore, instead of letting devices simply connect to the closest APs or base stations, a more intelligent load balancing mechanism can, for instance, take into account the amount of traffic already going over that particular infrastructure device. By taking into account the different characteristics of the devices (e.g., available technologies or paths) and technologies (e.g., bandwidth or range), the approach can cope with the heterogeneity across the network and account for the mobility and the growing number of devices, as highlighted in Section 1.2. Finally, through the deployment of novel monitoring functions this entity can have a more detailed insight in neighboring networks and detect traffic related patterns in order to optimize its own controlled network.

Summarized in layman's terms, we envision a system where a user should only be aware of the fact that its device is connected to the Internet, while the network takes care of all the underlying decision-making and provisioning of resources, as such offering an improved QoS and user experience. This idea can be formulated as **connectivity as a service**.

1.4 Research Questions

In order to validate the formulated hypothesis, we define four research questions. Each of these questions is targeting one of the stated problems in Section 1.2. Respectively, we have the following research questions:

1. **How can we enable seamless inter-technology management transparent to all actors?** From the previous sections, it is clear that there is a

strong need for reliable and seamless inter-technology management. A key observation is that this problem exists in all kinds of different networks and devices. In order to provide a generally applicable solution, the key requirement is the transparency towards individual technologies, end-users and their applications, and the network itself (e.g., other devices). Furthermore, management operations such as inter-technology handovers should take place in a seamless manner with no noticeable impact on the user experience or application behavior. While also the impact on the network should be mitigated through centralized control over all actors. However, introducing centralized control requires management communication channels between the controller and the enabled devices. Finally, in general, we need to make sure that the effort of deploying the envisioned solutions is kept minimal. Especially since a modification to all kinds of connected devices is considered, we need to carefully introduce this and make sure that legacy support and intermediate steps are available as well.

2. **Can intelligent routing of traffic streams significantly improve network performance?** The envisioned intelligent traffic routing is needed to cope with the stringent requirements of modern services. It could also allow increasing the available bandwidth that traffic flows can consume, without the need for lowering the offered quality of the traffic flows or introducing additional hardware (e.g., infrastructure devices like Wi-Fi APs) to the network. The main question is how significant the impact of such an approach on the network-wide performance will be. If this is only a few percentages, users will barely notice any differences when consuming services or using applications. Furthermore, the approach should be able to address issues or provide optimizations as fast as possible, based on the current state of the network. For instance, upon link failure, a fast and adequate response is needed to minimize the loss of performance and experience. Furthermore, as network environments can be very diverse, the approach should be independent of technologies and the different network management solutions that are in place to enable, among others, inter-technology handovers. Finally, in order to provide an adequate solution, the demands of traffic flows need to be known, while also the actual capacities of the technologies need to be taken into account. The latter is especially important and more challenging for wireless connections and technologies.

3. **Can the impact of mobility and the growing number of devices and technologies be countered by introducing intelligent load balancing?** As mobile devices are a key segment within the group of current and future connected devices, it is clear that network management needs to take the aspect of mobility into account. An intelligent approach should leverage different aspects, such as among others load, distance, and different demands in order to provide the best QoS possible by dividing devices and their traffic across various technologies and network paths. Moreover, in order to be generally applicable, this load balancing approach should be able to take into account

different unique mobility patterns of devices amongst the different networks. This on top of the already previously mentioned independence towards, for instance, technologies. Because of the fact that mobility can vary often, the importance of a responsive approach is even higher than stated in the previous research question. Within the scope of the ever-growing amounts of connected devices, it is clear that maintaining this responsiveness is critical. The envisioned load balancing approach needs to take this into account and scale along these rising numbers. In order to guarantee this, an investigation towards the trade-off between scalability and optimality should be conducted.

4. **Can we detect traffic patterns of neighboring networks using spectral data?** Because of the interference between neighboring networks and technologies, their coexistence is becoming non-trivial. To this extent, network management solutions should be upgraded with a novel monitoring mechanism that allows them to detect and mitigate this interference. Based on spectral data it is already possible to detect interference of external sources like neighboring technologies or devices such as a microwave. However, if it would be possible to detect traffic patterns within a certain technology like Wi-Fi, management systems could use this information to increase the performance of their network. Existing traffic recognition approaches operate on a packet-level, which typically requires that the listening device is part of the corresponding network. For instance, imagine a scenario wherein neighboring network traffic is being transmitted in a burst pattern of 0.3 s per interval of 1 s on a certain technology and channel. This means that no interference will occur when using the other 0.7 s of that interval for the transmission of traffic within the controlled network. When management systems are capable of detecting these interference-free periods, traffic can, for instance, be offloaded to these channels or technologies during these free slots. The key challenge will be to construct a model that is capable of detecting these traffic patterns at the spectral level, to enable the required monitoring capabilities without the need of being connected to the different networks under observation. In order to train this model, sufficient amounts of data need to be collected. Furthermore, such a model needs to be robust in order to cope with the very dynamic characteristics of the different wireless environments.

1.5 Research Contributions

This dissertation aims to improve the inter-technology management of wireless networks. In particular, we validate the previously stated hypothesis and investigate the four formulated research questions. To this extent, the following four main contributions can be identified:

1. **The ORCHESTRA framework for seamless inter-technology network management in heterogeneous networks** (Chapter 3)

    - Design of an inter-technology management framework that consists of two components: a Virtual MAC (VMAC) layer and a centralized controller.
    - The VMAC offers a single connection point to the upper layers, while transparently bonding over the underlying network technologies.
    - Introduction of packet-level inter-technology handovers, load balancing, and duplication across the different underlying links by using packet matching rules.
    - The centralized controller maintains a global network overview by receiving detailed monitoring information from each VMAC and can in return enforce instructions (i.e., propagate rule changes).
    - Support for gradual network-wide roll-out through the transparency towards standard (i.e., legacy) devices and the possibility to interact with existing network controllers.
    - Definition of a number of highly relevant use cases that are suited for fast adoption of the framework.
    - Design and implementation of a real-life prototype that is capable of working with Ethernet, Wi-Fi (in both the 2.4 GHz and 5 GHz bands), and LTE.
    - An in-depth evaluation, using the prototype setup, demonstrates the seamless inter-technology management and compares the performance of the listed features of ORCHESTRA to Multipath Transmission Control Protocol (MPTCP), the standard solution used by industry.

2. **A real-time dynamic flow management approach to optimize network-wide throughput** (Chapter 4)

    - Design of an Mixed Integer Linear Programming (MILP) formulation that calculates the optimal path configuration in order to maximize the throughput of all individual traffic flows across the network, taking into account TCP fairness mechanisms.
    - Design of an MILP formulation that calculates the optimal path configuration in order to maximize the throughput across all different collision groups (encapsulation of a group of interfering links or stations) within the network.
    - The flow management approach is constructed fully independent of the underlying network management tools (e.g., the previously introduced ORCHESTRA framework or a Software-Defined Networking (SDN)-based setup).

- The decision-making process is based only on real-time gathered monitoring data, allowing a fast response to dynamic network events.
- The specific nature of wireless networks (e.g., the impact of competing stations within the network) is considered. Moreover, a dynamic method is presented to estimate the actual capacity of wireless network technologies. Note that within the scope of this dissertation, we denote the capacity of a technology as the total amount of transmittable data at a given moment.
- Thorough evaluations across a wide range of scenarios using an NS-3 based framework show a promising increase with respect to a default static baseline.
- A small-scale prototype setup is used to demonstrate how the approach can be used to improve the performance of a video stream upon the presence of a large download transfer.

3. **A scalable device and traffic load balancing approach taking into account the mobility of the connected devices** (Chapter 5)

   - Design of an MILP formulation that optimizes the path configuration and station associations within wireless networks to maximize the overall network throughput.
   - Design of a near-optimal two-step heuristic approach that splits the overall problem and optimally solves the problems of station association and traffic scheduling separately. To this extent, two different MILP formulations for the respective problems are solved after each other.
   - Design of a greedy heuristic approach that also considers the problems of stations association and traffic scheduling individually, after one another.
   - These approaches operate fully independently of the underlying framework, based on only real-time monitoring information acquired from the network. As such, honoring the properties stated in the previous research contribution.
   - A dynamic fingerprinting-based approach is proposed to capture the effect of mobility and distance on the maximum possible data rate per station. In other words, take into account the reduction in achievable data rate due to the decrease in Modulation and Coding Scheme (MCS) values and the increase in distance.
   - Using an NS-3 based framework, the three algorithms are evaluated across an extensive series of varying scenarios. A significant improvement, up to more than 100 %, towards a static baseline is demonstrated.
   - A scalability analysis shows that by using a heuristic approach, wireless network management can be provided for networks up to 10000 devices.

4. **A Machine Learning (ML) model to detect traffic patterns on a spectral level** (Chapter 6)

   - Design of a ML approach that is capable of detecting traffic patterns (e.g., interval occupancy, specific protocols, or rates) on spectral data of uncontrolled neighboring networks.
   - A domain randomization-based approach is proposed that uses large-scale synthetic generated data to train the model, as such allowing for a more robust model cable of coping with the very dynamic and ever-changing nature of wireless environments.
   - A framework is constructed that combines two state-of-the-art simulators, namely the NS-3 simulator and the Matlab toolbox, in order to generate large amounts of synthetic data to train different ML models.
   - We present a Convolutional Neural Network (CNN) architecture that forms the basis for prediction models to recognize TCP and User Datagram Protocol (UDP) traffic, burst traffic with different duty cycles, and different transmission rates. These models are trained using the aforementioned synthetic data.
   - We explore and compare two different approaches to represent the sensed spectrum as the input of our models: time and time-frequency image representations. Furthermore, we show that all models, using both approaches, have an accuracy of more than 96 %, when being validated with a synthetic test dataset.
   - We present a validation with real-life data that indicate the applicability of a domain randomization approach in a wireless context. It is possible to detect different transmission rates with an accuracy of 86.9 %.
   - A small-scale prototype setup is presented to demonstrate how one of the proposed models can be used in a real-life setting.

In order to solve the challenge towards the management of heterogeneous wireless networks, the combination of the different research contributions is required. Figure 1.3 illustrates how the four contributions are aligned together. First of all, the ORCHESTRA framework is installed on all devices in the network, in order to enable the different multi-technology features such as seamless inter-technology handovers. For instance, within a LAN (as shown in the left side of Figure 1.3) the ORCHESTRA VMAC is installed on all devices, both consumer and infrastructure devices. The controller is placed somewhere in the infrastructure (e.g., on a modem or a separate connected device). Furthermore, the ORCHESTRA framework can also be deployed in a multi-technology backhaul network controlled by a network operator, as depicted at the right side of Figure 1.3. This allows to efficiently utilize the capacity of both wired (e.g., Digital Subscriber Line (DSL) or fiber) and wireless (e.g., Wi-Fi or LTE) networks. Research contributions 2 and 3 provide the intelligence needed on top of the management framework of the first research

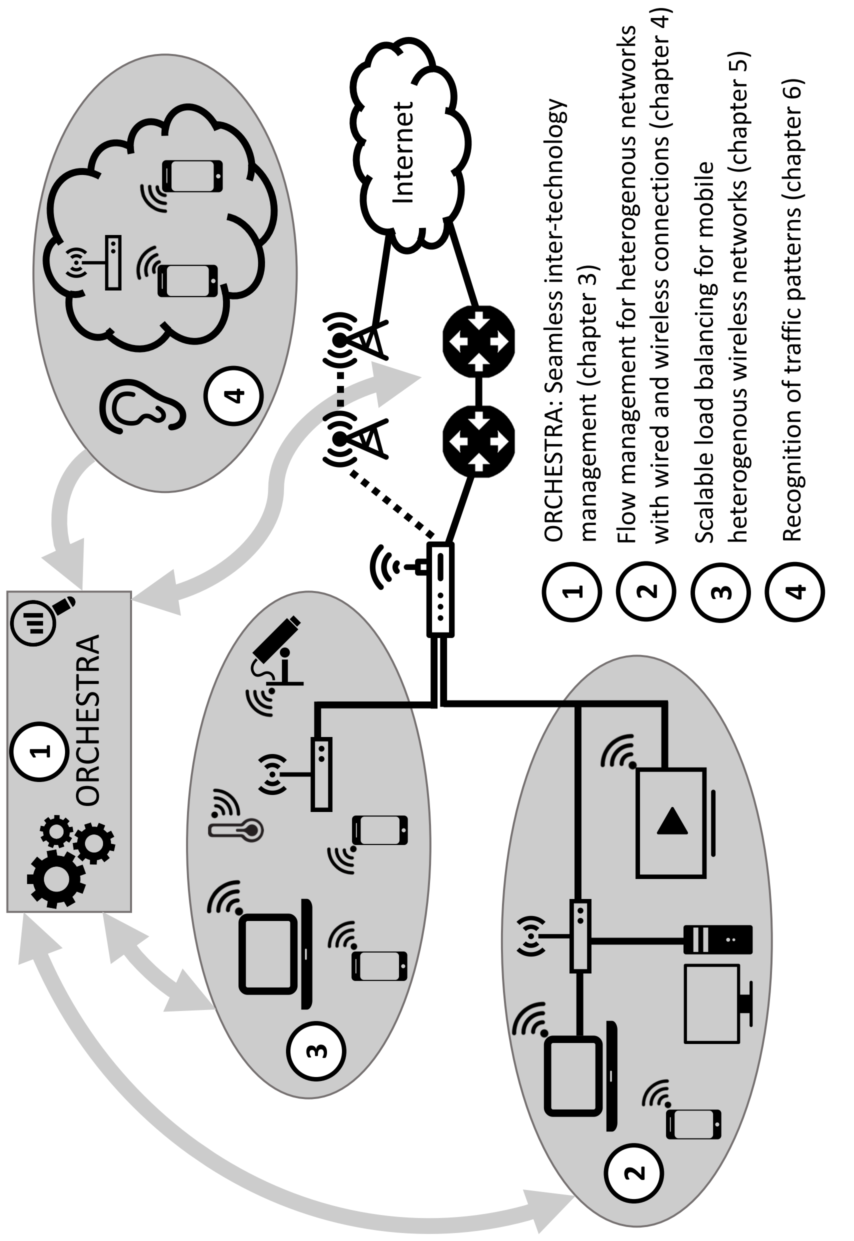

ORCHESTRA
1
2
3
4
Internet
1 ORCHESTRA: Seamless inter-technology management (chapter 3)
2 Flow management for heterogenous networks with wired and wireless connections (chapter 4)
3 Scalable load balancing for mobile heterogenous wireless networks (chapter 5)
4 Recognition of traffic patterns (chapter 6)

Figure 1.3: Overview of how the different research contribution are linked together and address the issue of heterogeneous wireless network management.

contribution, in order to optimize the network. The intelligence is installed on top of the ORCHESTRA controller and uses the framework to acquire real-time monitoring information and to roll-out policies across all devices (e.g., to perform an inter-technology handover). The bottom left area of Figure 1.3 visualizes a network, consisting of both wired and wireless technologies, that is managed by the work of the second research contribution. Similarly, the middle part of Figure 1.3 illustrates how research contribution 3 allows managing a network consisting of mobile wireless devices. Finally, the fourth research contribution allows wireless infrastructure devices to listen to the spectrum and identify traffic patterns of neighboring networks. This information can via the management framework be shared with the network management algorithms, for further exploitation and network optimization.

1.6 Dissertation outline

This dissertation is in total composed of seven main chapters. Following this introduction, Chapter 2 offers a detailed view of the state-of-the-art in the area of (wireless) network management. In particular, we discuss existing work in the areas of inter-technology management and handovers, load balancing, and traffic recognition. The next four chapters focus on different research contributions. Chapter 3 introduces the ORCHESTRA framework for seamless inter-technology network management (Research contribution 1). Chapter 4 provides the intelligence that can be run on top of the ORCHESTRA framework (Research contribution 2). In particular, the focus lays on the flow management and traffic steering part. This work is continued in Chapter 5 where the focus shifts towards the applicability of load balancing in mobile wireless networks (Research contribution 3). Furthermore, heuristics are provided to cope with the growing sizes of the networks of today and tomorrow. In Chapter 6 the possibility of detecting traffic patterns on the physical level of spectral data is explored (Research contribution 4). Finally, in Chapter 7, we draw conclusions and list possible future research related to all the different contributions of the dissertation.

1.7 Publications

The research results obtained during this PhD research have been published in scientific journals and presented at a series of international conferences. Furthermore, there is also a patent application submitted for the contributions regarding the ORCHESTRA framework. The following list provides an overview of the publications and patent applications during the PhD research.

1.7.1 A1: Journal publications indexed by the ISI Web of Science ”Science Citation Index Expanded”

1. **Tom De Schepper**, Steven Latré, and Jeroen Famaey. *Flow Management and Load Balancing in Dynamic Heterogeneous LANs.* Published in IEEE Transactions on Network and Service Management (TNSM), vol. 15, no. 2, pp. 693-706, June 2018. doi: 10.1109/TNSM.2018.2804578. [Impact factor: 3.286]

2. **Tom De Schepper**, Patrick Bosch, Ensar Zeljković, Farouk Mahfoudhi, Jetmir Haxhibeqiri, Jeroen Hoebeke, Jeroen Famaey, and Steven Latré. *ORCHESTRA: Enabling Inter-Technology Network Management in Heterogeneous Wireless Networks.* Published in IEEE Transactions on Network and Service Management (TNSM), vol. 15, no. 4, pp. 1733-1746, December 2018. doi: 10.1109/TNSM.2018.2866774. [Impact factor: 3.286]

3. **Tom De Schepper**, Steven Latré, and Jeroen Famaey. *Scalable Load Balancing and Flow Management in Dynamic Heterogeneous Wireless Networks.* Published in Journal of Network and Systems Management (JNSM), June 2019. doi: 10.1007/s10922-019-09502-2. [Impact factor: 1.750]

4. **Tom De Schepper**, Patrick Bosch, Ensar Zeljković, Jakob Struye, Carlos Donato, Farouk Mahfoudhi, Jeroen Famaey, and Steven Latré. *ORCHESTRA: Supercharging Wireless Backhaul Networks through Multi-technology Management.* Submitted to IEEE Journal on Selected Areas in Communications (JSAC), January 2019. [Impact factor: 7.172]

5. **Tom De Schepper**, Miguel Camelo, Jeroen Famaey, and Steven Latré. *Traffic recognition at the spectrum level using synthetically trained deep learning models.* Submitted to International Journal of Network Management (IJNM), August 2019. [Impact factor: 1.231]

6. Patrick Bosch, **Tom De Schepper**, Ensar Zeljković, Jeroen Famaey, and Steven Latré. *Orchestration of Heterogeneous Wireless Networks: State of the Art and Remaining Challenges.* Submitted to Computer Communications, July 2019. [Impact factor: 2.613]

1.7.2 P1: Proceedings included in the ISI Web of Science ”Conference Proceedings Citation Index - Science”

1. **Tom De Schepper**, Bart Braem, and Steven Latré. *A virtual reality-based multiplayer game using fine-grained localization.* In proceedings of the Global Information Infrastructure and Networking Symposium (GIIS), Guadalajara, Mexico, pp. 1-6, October, 2015. doi: 10.1109/GIIS.2015.7347176.

2. **Tom De Schepper**, Alexander Vanhulle, and Steven Latré. *Dynamic BLE-based fingerprinting for location-aware smart homes.* In proceedings of the

IEEE Symposium on Communications and Vehicular Technology (SCVT), Leuven, Belgium, pp. 1-6, November, 2017.
doi: 10.1109/SCVT.2017.8240316.

3. Ensar Zeljković, **Tom De Schepper**, Patrick Bosch, Ian Vermeulen, Jetmir Haxhibeqiri, Jeroen Hoebeke, Jeroen Famaey, and Steven Latré. *ORCHESTRA: virtualized and programmable orchestration of heterogeneous WLANs.* In proceedings of the International Conference on Network and Service Management (CNSM), Tokyo, Japan, pp. 1-9, November, 2017.
doi: 10.23919/CNSM.2017.8255999.

4. **Tom De Schepper**, Jakob Struye, Ensar Zeljković, Steven Latré, and Jeroen Famaey. *Software-Defined Multipath-TCP for Smart Mobile Devices.* In proceedings of the International Conference on Network and Service Management (CNSM), Tokyo, Japan, pp. 1-6, November, 2017.
doi: 10.23919/CNSM.2017.8256043.

5. Ian Vermeulen, Patrick Bosch, **Tom De Schepper**, and Steven Latré. *Di-Mob: Scalable and seamless mobility in SDN managed wireless networks.* In proceedings of the International Conference on Network and Service Management (CNSM), Tokyo, Japan, pp. 1-6, November, 2017.
doi: 10.23919/CNSM.2017.8256048.

6. Patrick Bosch, **Tom De Schepper**, Ensar Zeljković, Farouk Mahfoudhi, Yorick De Bock, Jeroen Famaey, and Steven Latré. *A demonstration of seamless inter-technology mobility in heterogeneous networks.* In proceedings of the IEEE International Symposium on a World of Wireless, Mobile, and Multimedia Networks (WoWMoM), Chania, Greece, pp. 1-3, June, 2018. doi: 10.1109/WoWMoM.2018.8449788.

7. **Tom De Schepper**, Steven Latré, and Jeroen Famaey. *Load balancing and flow management under user mobility in heterogeneous wireless networks.* In proceedings of the International Conference on Network and Service Management (CNSM), Rome, Italy, pp. 1-7, November, 2018.

8. Miguel Camelo, **Tom De Schepper**, Paola Soto, Johann Marquez-Barja, Jeroen Famaey and Steven Latré. *Detection of traffic patterns in the radio spectrum for cognitive wireless network management.* Submitted to the IEEE International Conference on Communications (ICC), August, 2019.

1.7.3 C1: Other publications in international conferences

1. **Tom De Schepper**, Steven Latré, and Jeroen Famaey. *A transparent load balancing algorithm for heterogeneous local area networks.* In proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), Lisbon, Portugal, pp. 160-168, May, 2017.
doi: 10.23919/INM.2017.7987276.

2. **Tom De Schepper**, Patrick Bosch, Ensar Zeljković, Koen De Schepper, Chris Hawinkel, Steven Latré, and Jeroen Famaey. *SDN-based transparent flow scheduling for heterogeneous wireless LANs.* In proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM), Lisbon, Portugal, pp. 901-902, May, 2017.
   doi: 10.23919/INM.2017.7987404.

1.7.4 Patent applications

1. Patrick Bosch, **Tom De Schepper**, Ensar Zeljković, Jeroen Famaey, and Steven Latré. *Network stack for a plurality of physical communication interfaces.* European patent application EP17171131.0, submitted May 2017.

2
State-Of-The-Art

"Don't ask me who's influenced me. A lion is made up of the lambs he's digested, and I've been reading all my life."

– Charles de Gaulle (1890 - 1970)

2.1 Introduction

In order to solve the challenges and problems in the complex and heterogeneous wireless networks of today and tomorrow, this dissertation introduces intelligent inter-technology management solutions. As the underlying problems in these heterogeneous environments are rather substantial and have a large impact on the performance of the network, the behavior of applications, and the experience of the end-user, efforts and innovations have already been presented within this domain. In this section, we discuss the relevant state-of-the-art in detail and aim to provide a clear overview of the missing features and aspects.

In particular, we first discuss existing multi-technology management solutions in Section 2.2. This corresponds to the first research contribution and the work presented in Chapter 3. Second, we focus on existing load balancing approaches within different domains in Section 2.3. This provides the starting point for the work covered by the second and third research contributions and their corresponding Chapters 4 and 5. Section 2.3 covers existing work focusing on traffic recognition, which corresponds to the fourth and final research contribution and the work proposed in Chapter 6. For each of these different areas covered, a clear summary is provided at the end of their respective sections.

2.2 Multi-technology control and management solutions

In the last decade, a number of approaches have been proposed to facilitate inter-technology handovers and multi-technology load balancing, the key features of the envisioned intelligent and seamless inter-technology management. In this Section, we first focus on highlighting the most relevant solutions. These works are presented by approximately following the order of the OSI stack, starting from the more technology-specific solutions, up to the higher layers. Afterwards, a summary is provided in Section 2.2.7, where the different solutions are compared to each other and the missing aspects and features are identified.

2.2.1 Media Independent Handover (IEEE 802.21)

Traditionally, efforts were made to streamline and facilitating the process of intra-technology handovers or roaming across different infrastructure devices within a single technology such as Wi-Fi or 3G/4G. For instance, the IEEE 802.11k, r, and v amendments for Wi-Fi. After the introduction of mobile devices, multi-homed devices started to appear that were enabled with multiple technologies and could as such connect to different networks. In order to introduce similar seamless mobility across those different networks (in particular between LANs and Wide Area Networks (WANs)), and to speed up mobile IP handovers, the Media Independent Handover (MIH) standard was proposed in 2007 [40,41]. This standard allows for the continuation of IP sessions across different technologies and networks, by the introduction of inter-layer messages that are exchanged through the Media Independent Handover Function (MIHF). However, in order to enable this, adaptations to the underlying technology standards are required, such as the introduction of a new link layer (called Service Access Point (SAP)) to handle the inter-layer messages. Communication between MIHFs of different wireless technologies is managed by event notifications, commands, and information services. An event notification can include a warning about dropping signal quality, while a command can be used to initiate a handover between technologies. Information services are used to exchange information between the different layers and the MIHF. The MIH standard can be used with various IP protocols, including Session Initiation Protocol (SIP) and Mobile IP, to facilitate handovers. The latter is, for instance, the case for Mobile IPv6 [42]. However, no guarantees on the handover (e.g., in terms of duration) are given [43]. Moreover, it has been shown that the execution time for the handovers is high in highly dynamic networks [43]. Another disadvantage of MIH, is that it requires a single legal network entity for each mobile node, in order to provide security [43]. The standard provides information to allow handovers to and from Ethernet, Wi-Fi, IEEE 802.16 Worldwide Interoperability for Microwave Access (WiMAX), IEEE 802.15, and 3G/4G networks. Amendments have been proposed to enhance security and authentication mechanisms [43,44].

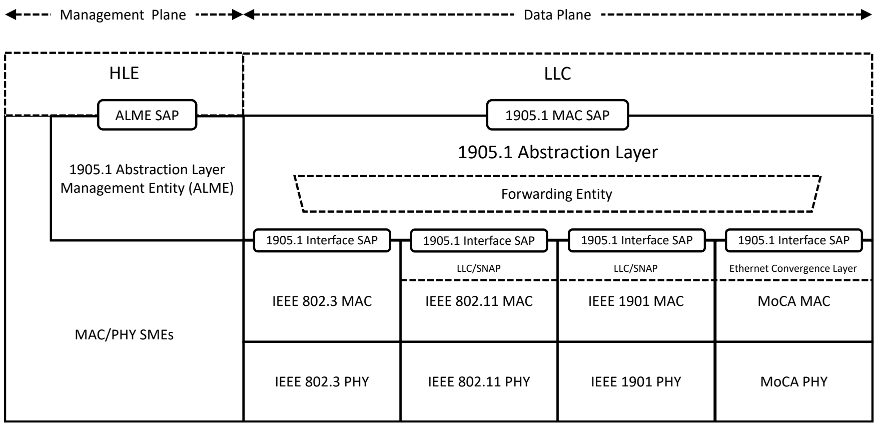


Figure 2.1: Overview of the introduced abstraction layer by the IEEE 1905.1 standard [45].

2.2.2 IEEE 1905.1 standard

A key aspect in terms of user-friendliness and experience is the abstraction from network connectivity, as users do not want to struggle with the low-level specifics of each network technology. One of the main focal points so far has been the development of a unified high bandwidth environment that exploits the multiple interchangeable available network technologies on most devices [46]. Earlier work towards a converged gigabit home network and an Inter-Mac architecture, eventually resulted in the definition of the IEEE 1905.1 standard in 2013 [45,47,48]. This architecture, as shown in Figure 2.1 introduces an abstract (or virtual) MAC layer on top of the current data link layer (i.e., OSI layer 2) to combine all the heterogeneous MAC interfaces in a transparent manner [49]. A unique virtual MAC address is assigned to represent each device on the network. Packet header matching rules can be used to transparently hand over flows and to load balance different flows across the different interfaces. As such, the implementation of the IEEE 1905.1 standard results in a simplified set-up, configuration, and operation of network devices with heterogeneous technologies and allows for dynamic rerouting of flows. The standard targets in particular LANs as it supports Ethernet, Wi-Fi, (Powerline) HomePlug (HP), and Multimedia over Coax (MoCA). Despite its potential, IEEE 1905 was never really adopted by industry, and only a few products exist that support it (e.g., Qualcomm Hy-Fi). Since its release in 2013, no follow-up releases or developments have been proposed.

2.2.3 Software-Defined Networking-based approaches

An alternative for using modified data link layers or a hybrid MAC layer can be found in the popular SDN principle, especially as it is being introduced in LANs [39, 50]. The separation between control and data plane, and the notion of network programmability, is typically enabled through the OpenFlow communication protocol that gives access to the different infrastructure devices in the network. The installation of an SDN-enabled switch, like Open vSwitch (OVS), makes it possible to perform transparent handovers between the different available technologies, such as Ethernet, 2.4 GHz Wi-Fi, and 5 GHz Wi-Fi [51]. An OpenFlow controller (e.g., Ryu or Floodlight) is used to query the OpenFlow-enabled devices for real-time monitoring information to identify different data flows and their requirements by the exchange of OpenFlow stats request and reply messages [51,52]. Link quality information can be acquired in a similar fashion. Furthermore, the controller can initiate the handover of a subset of flows to another technology by changing the outgoing OpenFlow port.

Recently, the focus of research has expanded towards the management of the network's wireless segment [26, 53]. To this extent, ODIN is one of the first wireless SDN controllers [53, 54]. Its goal is to make dense wireless networks more manageable, provide a smooth Wi-Fi experience to users, and support QoS for a wide array of applications and use cases. Key in its design is the introduction of the Light Virtual AP (LVAP) abstraction, as an addition to the default virtualization of APs (i.e., Virtual APs (VAPs)). This concept virtualizes the association state and separates this from the physical AP. Stations will now connect to their unique LVAP instead of the underlying physical AP. This allows for the seamless mobility of stations as they will remain associated, and only the corresponding LVAPs are transferred to other physical APs. The ODIN architecture consists of two parts: the ODIN master (i.e., controller) and the ODIN agent running on the physical APs (using OpenWRT). The ODIN master is implemented on top of the Floodlight OpenFlow controller. As such, it supports full OpenFlow capabilities and maintains a global view over the network, including the status of APs, stations, and OpenFlow switches. Furthermore, ODIN employs the principle of a split-MAC where time-critical operations (e.g., transmission of Acknowledgements (ACKs)) are performed by APs, and non-time-critical operations (e.g., station association) are handled by the controller [26, 54].

A recent SDN-based wireless orchestration architecture is the 5G-EmPOWER networking framework [55, 56]. It is inspired by, and builds further on top of, the principles of the previously mentioned ODIN framework [26]. In particular, 5G-EmPOWER also uses the principle of LVAPs in order to manage the mobility of stations. Remaining faithful to the SDN principle, 5G-EmPOWER moves the intelligence away from the infrastructure devices (e.g., Wi-Fi APs) to a (centralized) controller. However, compared to ODIN, it extends the programmability of the network through either a number of Python interfaces or a REST Application Programming Interface (API) and offers an increased amount of Virtualized Network Functions (VNFs) [26, 55]. As such, it offers increased control and insight in the available resources in the network (e.g., available bandwidth or load per physical AP). Currently, its focus is on the following control aspects: wireless clients state management, resource allocation, network monitoring, and network reconfiguration [26, 55]. These features can be exploited, through the northbound interfaces, by applications running on top of the controller that get their own network slice (i.e., virtual network) assigned [55, 56]. Finally, it is key to highlight that the offered functionalities are not only available for Wi-Fi networks but that there is also support for cellular networks (currently 3G/4G, but in the future also 5G technologies) and devices (virtualization of LTE Evolved Node B (eNB)) [55].

As wireless SDN controllers and frameworks have become a popular research topic, a multitude of other solutions have been proposed. Dezfouli et al. present a recent and extensive overview of these existing solutions [26]. Most of the listed works reuse or reformulate previously mentioned ideas (such as VAPs, LVAPs, or the notion of a split-MAC) while focusing on specific use-cases or features. Notable examples are, for instance, Ethanol that claims that the standard OpenFlow protocol is not suited to provide QoS in wireless networks or Virtualized Access Network (VAN) that targets residential environments and allows to share bandwidth with neighbors [26, 57, 58]. Similarly, the European Horizon 2020 Wi-5 project employs these principles with the goal to manage Wi-Fi APs more efficiently, offering inter-provider cooperation and interference mitigation [59]. Despite the large amount of existing work, remaining challenges are, among others, coping with the coexistence of overlapping channels and networks, managing dense networks, offering scalable solutions, and supporting high-throughput or legacy clients [26].

2.2.4 3GPP and Tunneling approaches

In order to cope with the ever-growing bandwidth and traffic speed demands, especially towards the highly hyped 5G networks, the Third Generation Partnership Project (3GPP) community began exploring the wireless spectrum outside of the traditional licensed 3G/4G bands. Two different approaches have been proposed to offload traffic from the cellular networks: first, the use of unlicensed spectrum (i.e., LTE-Unlicensed (LTE-U) and LTE License Assisted Access (LTE-LAA)) and, second, the addition and use of Wi-Fi technologies (i.e., LTE-WLAN Aggregation (LWA)) [60–62]. In the first case, LTE is directly used in the unlicensed spectrum (specifically the 5 GHz band). However, this can potentially cause severe performance degradations in coexisting Wi-Fi systems [19,27]. Furthermore, note that different LTE-U deployments of different operators can also interfere with each other [63]. In contrast to LTE-U, LTE-LAA contains a Listen-Before-Talk (LBT) protocol and employs a so-called freeze period, where LTE leaves free airtime for other technologies. This allows LTE-LAA to be used on a larger scale and provides better coexistence with, for instance, Wi-Fi technologies. It has been shown that the throughput per Wi-Fi AP, under coexistence with LTE-LAA, is comparable to cases where the AP shares its spectrum with other IEEE 802.11 devices. LTE-LAA has been standardized in 3GPP Release 13 (downlink traffic) and Release 14 (uplink traffic and dynamic channel selection).

On the other hand, LTE-LWA proposes to combine an LTE eNB with one or more Wi-Fi APs by either a physical integration or an external network interface [62,64]. While the 5 GHz band will still be more heavily utilized, the LWA approach introduces fewer coexistence issues, and no hardware changes are required on the infrastructure [27]. From a user perspective, both LTE and Wi-Fi are used seamlessly as mobile traffic flows are tunneled over the Wi-Fi connection and can be handed over between both technologies. Current research for LWA focuses mainly on achieving high performance and low latency handovers. This is, for instance, achieved by decreasing the overhead of handovers and scheduling them properly, which leads to a reduced handover duration [65,66]. Currently, only two LWA deployments are planned worldwide (in Singapore and Taiwan), while already over 30 trials and deployments (both planned or launched) exist for LTE-U and LTE-LAA [67].

Outside of the 3GPP community, the MulteFire Alliance tries to fill the market for small cells and local deployment, also known as private LTE [68,69]. The first version of MulteFire was specified in 2017 and is based on LTE-LAA. It supports an LBT protocol, as well as private deployments and mainly works in unlicensed and shared spectrum, in particular the 5 GHz and the 3.5 GHz (i.e., CBRS) frequency bands. Contrary to standard LTE deployments, no service provider is necessary, but it can be connected to a public network as a neutral host. As such, deployment and operation is similar to simplicity offered by IEEE 802.11 deployments. While currently no cooperation with other technologies is considered, it is very likely that MulteFire products will be roll-out in parallel with upcoming Wi-Fi technologies (e.g., IEEE 802.11ax).

Furthermore, in light of the ongoing roll-out of 5G technologies, the 3GPP community issued Release 15 in 2018. This release, also informally called 5G phase 1, introduced the first 5G standards that specify, among others, the New Radio (NR) idea [70,71]. NR is a novel radio interface that eventually will replace the existing 3G/4G technologies, and as such, also the LWA, LTE-LAA, and LTE-U technologies. In contrast to these previous technologies, NR will support from the start operation in all frequencies from below 1 GHz up to 52.6 GHz [71]. Key in this will be the support for the frequencies above 6 GHz, as such introducing Millimeter-Wave (mmWave) communications to 5G, in order to find free spectrum to support massive bandwidth and high throughput requirements [70]. mmWave communications rely on beams between multiple directed antennas and Multiple-input and Multiple-output (MIMO) to offer Gigabit connections. However, critical elements are, among others, beamforming and the interworking (e.g., handovers) between the higher and lower frequencies [70,71]. Such features are currently still under (further) development, as NR is only at the beginning stages of development [70].

Finally, some other commercially available products exist that use a similar tunneling approach to hide away the underlying communication technologies. These products typically target LANs, and in particular office environments. A tunnel is configured between a so-called pro-active router or modem and an instance in the cloud, while the different technologies under the hood (e.g., DSL, fiber, satellite or LTE) are concealed. The router decides which underlying technology to use per traffic flow, based on QoS parameters. This technique is also known as Software-Defined Wide Area Networks (SD-WAN) bonding. It is, among others, offered by the companies Mushroom Networks and Peplink [72,73].

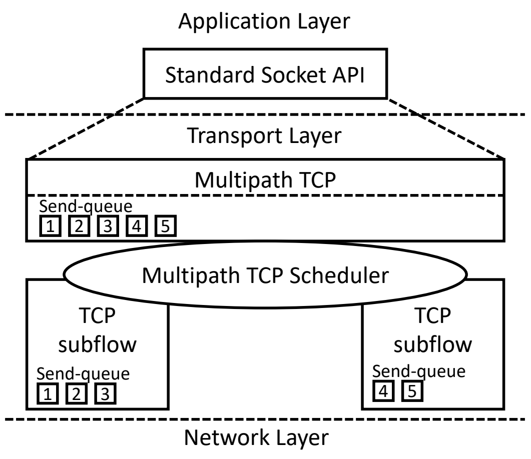


Figure 2.2: Architecture of the MPTCP stack with a prominent role for the scheduler [74].

2.2.5 Multipath Transmission Control Protocol

In 2013, the MPTCP standard was released as an extension to regular TCP [75]. This extension enables the transmission and reception of data concurrently on multiple network interfaces in order to maximize resource usage and increase redundancy in multi-technology networks [75, 76]. Multiple regular TCP connections (denoted as subflows), are offered as one to the application layer, while under the hood each subflow can follow different paths through the network. Based on the ever-changing network characteristics (e.g., increased Round Trip Time (RTT)), the MPTCP scheduler can divide or duplicate application data across these subflows to attain a higher throughput or increased reliability [74]. The concept of MPTCP is illustrated by Figure 2.2. Additionally, one sub-flow can be kept idle and serve as a back-up stream in case the main sub-flow would break or to use as a channel to send retransmissions. In this case, the fallback sub-flow is already established, meaning the handover can occur very quickly and fully transparent to upper layers. However, research has shown that these handovers are not seamless and can take up to 2-3 s [77, 78]

Moreover, as MPTCP is fully backwards-compatible with regular TCP, an MPTCP-aware host attempts to use MPTCP when establishing a new connection, but falls back to regular TCP gracefully when the other endpoint does not indicate it is MPTCP-aware. The key component in the whole MPTCP architecture is the scheduler that must decide which subflow(s) will be used to send each TCP segment [74, 79]. The most used scheduler, part of the default MPTCP implementation, is the Lowest Round Trip Time First (LowRTT) scheduler that selects the subflow with the lowest RTT when a segment is to be scheduled. Once a subflow is chosen, all the following segments are also sent using that subflow, until its congestion window is filled [74]. However, it is important to note that this scheduling is done per connection between two hosts and not on a network-wide scale.

Besides the scheduler, it has been shown that the maximum buffer size can also have significant impact on the performance of MPTCP [79–82]. Having a limited buffer size can lead to significant throughput degradations in heterogeneous network environments, especially when links have different characteristics [80, 81]. The latter is especially true for heterogeneous wireless environments where it has been demonstrated that the overall performance of MPTCP can be worse than regular TCP [79, 82]. Furthermore, MPTCP has proven to be very aggressive towards other non-controlled TCP connections in the network, sometimes even without providing added benefits for the MPTCP users [83].

Originally, MPTCP was designed with multi-homed devices such as smartphones (enabled with both Wi-Fi and mobile interfaces) or servers (that are set up with multiple Ethernet interfaces) in mind [75]. Currently, indeed, MPTCP is actively being used on a large scale in Android and iOS devices (e.g., by Siri) [84]. However, multiple telecommunication operators also employ MPTCP to split traffic across both wired and wireless backbone networks (called hybrid access networks). This is, in particular, the case for DSL and LTE solutions, to circumvent the limited capacity of DSL wires (also known as DSL-LTE bonding or hybrid-DSL). Such a solution is, among others, offered by the company Tessares [85]. Furthermore, MPTCP is also used to facilitate the transmission of video recordings from remote television crews in the field to the broadcasting headquarter, using a combination of satellite, 3G/4G, and Wi-Fi uplinks [86].

Finally, as the name suggests, MPTCP only supports TCP traffic. While it is possible to tunnel UDP packets over TCP, this introduces additional overhead to the already saturated wireless environments. Internet traffic is currently still dominated by TCP traffic, however, its share is decreasing as new applications and protocols are being launched that use UDP to reduce the network overhead [87, 88]. Examples of this are, the introduction of large scale M2M communications due to the adoption of IoT and the introduction of the UDP-based QUIC protocol by Google. To this extent, a multipath version of QUIC has recently been proposed [89].

2.2.6 Other solutions

So far, the most solutions discussed are located at the network or transport layers of the network stack. However, some solutions positioned at the application layer, are also available. First of all, the Border Gateway Protocol (BGP) is a decentralized routing protocol that has been around since the beginning days of the Internet [90]. It is based on TCP and used for routing between different autonomous networks and domains [90, 91]. Each (routing) device opens a TCP port and listens, as well as sends, keep-alive messages. As such a connectivity graph can be constructed that shows reachability between different networks and devices. This makes it possible to make routing decisions based on the available connections, enforce network policies, or configure network rules [91, 92]. While BGP is most known for its use in the routing of the core networks of the Internet (e.g., between different Internet Service Provider (ISPs)), it can be used for inter-technology or intra-technology handovers in smaller autonomous networks, such as wireless networks, as well.

SIP on the other hand, especially with its extensions, focuses on session mobility across different connections [93]. Each device is registered at a registrar which manages the current reachability of the device through its identifier. When the network or technology changes, the devices updates its IP address with its registrar, which in turn forwards it to registrars of currently connected devices. This allows for fast handovers, but there is a short downtime until the IP address is updated. SIP is currently used by Voice over LTE (VoLTE) to allow for voice calls over the mobile data connection [94].

Finally, it should be noted that recent OSs also focus on cross-technology integrations and bonding. First of all, this is the case for platforms targeting mobile devices such as smartphones or tablets. For instance, the latest smartphones can near seamlessly hand over connections between technologies, by closely monitoring Wi-Fi and LTE parameters. This is mainly achieved by reacting early on and preferring the more stable technology. This feature (integrated in the OS protocol stack) is, among others, known as Wi-Fi assist [95]. Second, the Linux kernel contains a module that offers a method of bonding (i.e., joining) two or more physical interfaces into one virtual interface [96]. The exact behavior can be specified through several available policies [97]. Examples of these policies are round-robin load balancing, backup (only 1 interface is used, while the others are kept as backups), or broadcast (all packets are transmitted over all interfaces). The feature originates from 2000, with the latest update in 2011 [96]. It is designed to increase the throughput or redundancy across two or more Ethernet links but requires support at both endpoints. As such, the main use cases are found in data centers or the wired backbone, although some enthusiasts have also proposed this in the context of Network-attached Storage (NAS).

2.2.7 Comparison and summary

In the previous sections, we have presented the most relevant multi-technology control and management solutions that operate on different layers of the OSI stack. Table 2.1 positions the different solutions next to each other and compares different features such as network domains, supported technologies, level of control, and supported transport protocols.

In general, we can say that nearly all listed approaches are technology dependent and/or target a specific network domain or use case. LANs are targeted by nearly all discussed technologies, except for the cellular ones. MPTCP and SDN solutions were originally designed for wired networks (e.g., in data centers, or in the core network) but are now also being applied to home and office networks. In terms of communication technologies, nearly all listed solutions support multiple communication technologies, with Wi-Fi being the most popular one. The exceptions are, in the first place, once again the 3GPP solutions, although LTE-LWA does combine LTE with Wi-Fi. Some older approaches, like BGP and the network bonding capabilities of the Linux kernel, target only Ethernet links.

Features	**IEEE 802.21**	**IEEE 1905.1**	**SDN-based**	**LTE-LWA**	**MPTCP**	**Application**
Network domains	LAN-WAN	LAN	LAN	LAN-Radio Access Network (RAN)	Any (end-to-end)	Any
Technologies	3GPP , Wi-Fi, IEEE 802.16,	Ethernet, HP, Wi-Fi, MoCA	Wi-Fi, 3GPP	Wi-Fi, LTE	All	All
Coordination	None	Global	Global	Local (within cell)	Between end-points	Local
Control-level	Flow-based	Flow-based	Flow-based	Flow-based	Packet-based (sub-flows)	Flow-based
Transport protocols	Any	Any	Any	Any	Only TCP	Any
Backward compatibility	Yes	No	No	Yes	Yes	Yes
Vertical Handovers	Yes	Yes	Yes	Yes (within cell)	Yes (between sub-flows)	Yes
Needs client changes	Yes (standards)	Yes	No	Yes	Yes	No
Products available	No	Qualcomm Hy-fi	Odin, 5G Em-POWER, ...	Two planned deployments	Android, iOS, Tessares, ...	VoLTE

Table 2.1: Comparison of existing multi-technology control and management solutions.

Furthermore, the above listed approaches operate in general on a flow-level, being able to reroute or hand over traffic flows across different network paths or connections. The major exception to all of this is MPTCP. MPTCP has already been applied in multiple domains and use cases, while being fully independent of the underlying communication technologies. It is also the most fine-grained solution that exists, as it allows to set up multiple sub-flows that can be used to transport the individual packets of a single traffic flow. However, MPTCP comes with two major drawbacks as it only supports TCP traffic and coordination is only possible between two endpoints and not network-wide.

This important network-wide coordination is currently only offered by the IEEE 1905.1 standard and SDN approaches (e.g., ODIN or 5G-EmPOWER) as they introduce a centralized controller to the network. Other approaches, such as LTE-LWA and application layer protocols, have a more local and distributed form of coordination by exchanging messages between the different involved devices. However, on the other hand, the IEEE 1905.1 standard and SDN approaches do not support legacy devices by default, while for instance LTE-LWA or MPTCP can easier fall back to, respectively, previous releases or standard TCP. Moreover, MPTCP, LTE-LWA, and IEEE 1905.1 require changes to the consumer devices, typically through a software update. Furthermore, although all listed approaches introduce intra- and inter-technology handovers, the seamlessness and performance of these operations can differ significantly. Typically, no guarantees can be provided on the duration of the handover. Finally, MPTCP is the only approach that, in addition to the handovers, also introduces features like duplication or packet-based load balancing. It should also be noted that both the IEEE 802.21 and IEEE 1905.1 standard have never really been adopted by industry, in stark contrast to, for instance, MPTCP.

Summarized, it is clear that the existing approaches fail to address the multi-technology problem in a fundamental manner, without targeting specific communication technologies or application domains. Furthermore, in order to boost network-wide performance there is a need for centralized coordination, accompanied by more fine-grained control, to also support packet-level operations and not only flow-level operations. The most relevant approach is MPTCP, followed by the IEEE 1905.1 standard, SDN-based solutions, and LTE-LWA.

2.3 Multi-technology load balancing approaches

All the approaches listed in the previous Section 2.2 define to some extent the features to enable multi-technology network management operations (e.g., handovers or load balancing). However, they do not define the intelligence or algorithms to actually optimize the network, such as selecting the most suitable path(s) per flow. In this dissertation, we mostly focus on intelligent load balancing. Within the context of this work, load balancing is considered to be the principle of balancing the effort of communication (between two or more devices), as much as possible, across all available components of the network infrastructure (e.g., nodes or links).

This is especially the case in the context of limited capacities and high demands. For instance, if multiple APs or base stations are present, the connecting devices should be balanced across all present infrastructure devices, and not all devices should be connected to the same infrastructure devices. Similarly, if devices are equipped with multiple communication technologies, not all devices should use the same technology, especially if the capacity is insufficient.

To this extent, we discuss existing load balancing algorithms in the areas of LANs and the so-called Heterogeneous Networks (HetNets) or RANs, respectively in Subsections 2.3.1 and 2.3.2. While load balancing techniques have also been applied in other types of networks (e.g., cloud networks) and domains (e.g., queuing), LANs and HetNets are the most relevant areas. Afterwards, a short summary is provided in Section 2.3.3.

2.3.1 Load balancing in heterogeneous local area networks

A number of contributions have been made that propose load balancing between wired and wireless links or between different wireless links mutually to cope with the increased traffic volumes in LANs. First of all, a per-flow decentralized load balancing technique is proposed by Sahaly and Christin, as part of a framework for heterogeneous home networks [98]. It is capable of reactively distributing incoming flows on the available links. However, the load balancing technique only takes local parameters per device into account and only a theoretical description is given without any real-life results. Furthermore, Macone et al. present a per-packet load balancing algorithm [49]. Per-packet load balancing can better exploit the network resources and thus theoretically provides better results. However, per-packet load balancing in combination with TCP can result in unnecessary retransmissions, due to the out-of-order arrival of packets. This results in severe throughput fluctuations in real-life systems when combined with standard TCP protocols. Furthermore, the algorithm runs centralized on the gateway and assumes full instantaneous knowledge of network resources and conditions.

Another decentralized load balancing algorithm, specifically for heterogeneous wireless access networks, is proposed by Oddi et al. [99]. This algorithm relies on a multi-connection transport layer in order to cope with the drawbacks of per-packet load balancing in the case of TCP. The proposed algorithm is based on the Wardrop equilibrium and does not take into account the fact that users do not have dedicated wireless network resources and are subject to contention and interference. In general, Olevera-Irigoyen has shown that determining the actual available bandwidth on the links has a big impact on the results of load balancing the flows in a (wireless) network, in particular with the time-varying capacity of Wi-Fi and Power line communication (PLC) [32]. Additionally, there are also load balancing solutions for LANs that focus on energy optimization by, for instance, selecting the most energy efficient link while still providing a good QoS [100, 101]. However, this is done by assuming the energy consumption model is known in advance, and not by real-time measurements.

While the previously listed approaches focus on balancing and scheduling across different technologies, proposals have also been made towards load balancing across different infrastructure devices within a single technology. The typical application is balancing the load of associated stations across multiple Wi-Fi APs. Popular approaches to tackle this problem are game theory and mathematical programming formulations [102–104]. For instance, Yen et al. show that a Nash equilibrium exists, and overall fairness and bandwidth are improved, in a game where stations greedily select an AP purely to maximize their own achievable throughput [102]. A more general game setup, taking into account the resources of different operators, is proposed by Malanchini et al., while using mathematical programming to optimally solve the game [103]. Similarly, a Mixed Integer Non Linear Programming (MINLP) formulation, taking into account the differences among the bandwidth demand of the different stations, has also been proposed [104]. Furthermore, Coronado et al. present a station association approach, that first assigns different channels to different APs in order to mitigate interference and collisions [105].

2.3.2 Load balancing in HetNets and mobile (LTE) networks

In HetNets, most research proposes technology-specific solutions that are capable of load balancing or performing handovers across only two technologies [106]. In particular, between either LTE and Wi-Fi or between Wi-Fi and WiMAX. The load balancing decisions are typically made centrally on the base station but sometimes also on a separate controller. Commonly, load balancing policies are based on the number of connected devices to a base station. Furthermore, a number of decision strategies have been proposed, among others, using utility functions, multiple attributes decision-making, Markov chains, and game theory [106–109]. For instance, a fully distributed algorithm based on the Nash equilibrium, for fair station assignments across Wi-Fi and WiMAX, is introduced by Coucheney et al. [110]. While Ye et al. propose a distributed dual decomposition-based algorithm, relaxing physical constraints, to provide a near-optimal solution for an optimal logarithmic utility maximization problem for equal resource allocation [111]. More recently, Harutyunyan et al. introduce an Integer Linear Programming (ILP) formulation for traffic-aware balancing devices across LTE and Wi-Fi infrastructures [112]. Note that this latter work, partially, extends the work presented in Chapter 4 by focusing on station association in RAN. Very recently, in light of the proposed NR principle for 5G networks interest has grown in handover and load balancing approaches for mmWave communications [70, 71]. For instance, a user association scheme based on MINLP has been proposed [113]. However, further research and optimizations are needed within this specific area [70].

In general, the listed strategies take only a limited number of parameters into account, with Received Signal Strength Indicator (RSSI) and Signal-to-noise Ratio (SNR) being the most popular ones [108, 114]. Open issues include, among others, the development of more generic solutions, better support for mobility, the use of multi-criteria decision functions, supporting different QoS classes, coping with asymmetric characteristics of downlinks and uplinks, and taking into account the capacity of backhaul links and User Equipment (UE) [108,109,115]. To conclude, current solutions are technology specific and do not take actual application or QoS parameters and objectives into account, making them unsuitable for use with QoS-sensitive or mission-critical services.

2.3.3 Summary

To summarize, most existing work on load balancing in heterogeneous networks focus on the development of theoretical models that are unusable in real environments as they assume the detailed knowledge of flow throughput requirements and dynamic network conditions. The specific nature of wireless networks (e.g., interference, link quality variability) and the typical behavior of TCP traffic flows are also ignored. Furthermore, developed algorithms tend to be technology specific (e.g., only operate on Wi-Fi or WiMAX) and are not suited for matching QoS requirements of modern services and end-users.

2.4 Traffic recognition approaches

In order to increase the spectrum efficiency and to cope with neighboring networks, we propose to recognize traffic patterns at the level of wireless radio signals. However, traditionally, traffic recognition takes place at higher layers of the network stack. To this extent, we discuss these traditional methods in Section 2.4.1. Next, in Section 2.4.2, we take a look at the efforts made in the area of Cognitive Radio (CR). Finally, we also provide a summary in Section 2.4.3.

2.4.1 Traditional traffic recognition approaches

Traffic recognition commonly takes place at gateways or routers in a (wired) network aiming to identify individual traffic streams or applications entering or exiting the controlled network environment. The typical use cases are network management (e.g., providing QoS or billing) by ISPs and intrusion or anomaly detection in the area of network security [116–118]. A very straightforward approach of traffic recognition is the association of port numbers with applications [117,118]. Combining this information with the MAC and IP addresses of the source and destination of the traffic flow, allows to identify individual applications, flows, and users. Despite its strong inaccuracy (e.g., multiple applications using the same port or port and address translations), this approach is still often used in practice, because it is simple to deploy and provides very fast continuous monitoring [118].

Two more advanced approaches exist: Deep Packet Inspection (DPI) and methods based on (IP) packet traces. Historically, DPI has long been the default approach where information is extracted from the headers and payload of individual packets, typically in a rule-based system. Especially towards network security, it can take place at different layers of the network stack (varying from the MAC to application layer) [117]. While DPI methods are very accurate, they require a lot of computational power, are very intrusive (i.e., towards privacy), and often require manual signature maintenance [116]. The intrusive nature of DPI is cumbersome in light of the recent global focus on data transparency and privacy regulations (e.g., GDPR). Furthermore, DPI can not always cope with the encryption of many modern end-to-end services. To this extent, recent work leverages the trade-off between encryption and privacy on one hand, and the functionalities and information exposed by DPI on the other hand [119]. The authors propose a set of novel protocols and encryption schemes that make it possible to perform DPI directly on the encrypted traffic, without compromising the privacy of the captured data.

As an alternative to DPI, statistical and ML-based methods have been proposed that do not require packet inspections but are based on captured packet traces, typically at the IP level [116, 120, 121]. These methods are based on traffic flow statistics like packet sizes, flow durations, inter-packet times, source and destination ports, and IP addresses [116, 122]. Both clustering (e.g., Expectation Maximization or K-Means), supervised learning (e.g., Naive Bayes or Genetic Algorithms), and semi-supervised learning (e.g., a combination of clustering and label mapping) techniques have been applied previously [122]. For instance, semi-supervised learning has been proposed to cope with zero-day applications, previously unknown to the traffic classification systems [123]. Overall, these methods can handle encrypted traffic, offer a lower computational cost than DPI methods, while still acquiring a rather high accuracy for flow classification (up to 99 %) [116, 122]. Lately, deep learning approaches have also been proposed [120, 124]. Lotfollahi et al. propose a combination of a stacked autoencoder with 5 fully connected layers and a one-dimensional CNN with as input IP layer packets [120]. The approach can identify applications with an accuracy of 98 % and traffic characteristics with a precision of 93 %. Furthermore, a CNN architecture is also proposed to detect malware traffic [124]. The model is trained on images, that are acquired by converting packets (pcap to idx format). Open issues include, among others, the questionable performance under non-perfect circumstances (e.g., packet loss, jitter, and packet fragmentation), while also only a limited amount of accurately labeled training data sets are available [116, 122]. Note that these available datasets only include data captured on wired links.

2.4.2 Cognitive radio approaches

The principle of CR has been introduced, comprising a large number of different approaches, to allow the coexistence of different technologies (both licensed and unlicensed) and networks over the same spectrum [36]. Within this domain of CR, the detection and classification of wireless radio signals are important to optimize spectrum usage. So far, the focus has been mostly on the recognition of modulation schemes and technologies. The detection of different technologies is typically done by exploiting key differences in, for instance, the channel access methods of the different technologies [125, 126]. Typical methods that have been applied are likelihood-based and feature-based, using high-order statistics features such as moments, cumulants and cyclic cumulants. For instance, Shi et al. propose a method to differentiate single carrier from Orthogonal Frequency Division Multiplexing (OFDM) signals, using the fourth order cumulants as features [125]. Karami et al. show how to discriminate between spatial multiplexing OFDM and Alamouti-coded OFDM for MIMO systems by relying on the second-order signal cyclostationarity [126]. Furthermore, they are also capable of distinguishing between GSM and LTE transmissions [127]. Finally, Liu et al. present an approach that uses the differences in RSSI distribution at Sub-Nyquist sampling rate to successfully recognize different technologies (e.g., Wi-Fi or LTE) [128].

Recently, deep learning techniques have gained momentum and are applied to classifying wireless signals. Schmidt et al. apply a CNN architecture to discriminate, with an accuracy of 95 %, between 19 different variants of modulation types and symbol rates within the 2.4 GHz band [129]. Similarly, Jeong et al. also apply a CNN to recognize different modulation types based on spectrograms, while Rajendran et al. apply a Long Short-Term Memory (LSTM) model that can classify 11 modulation types with an accuracy of 90 % [130, 131]. Both O'Shea et al. and Kulin et al. show how CNNs significantly outperform expert learning systems for the task of radio signal classification [132, 133]. They both compare multiple differently structured CNNs that are trained on processed In-phase and Quadrature (IQ) samples.

Within the context of CR, limited research exists that focuses on traffic recognition. First of all, a CNN has been developed to detect different interfering sources for an IEEE 802.15.4 sensor network, using RSSI traces [134]. The approach allows to detect with an accuracy of 93 % Wi-Fi beacons, Wi-Fi video streaming, Wi-Fi file transfer, iBeacon, and microwave oven interference in the 2.4 GHz band. Lately, Testi et al. have also shown that is it possible to use ML techniques to distinguish YouTube form WhatsApp traffic in a real-life setting [135]. The best results were achieved with a Neural Network (NN), achieving an accuracy of above 90 % under different SNR levels.

2.4.3 Summary

We have shown that traffic recognition is historically always performed at the higher layers of the network stack by DPI and methods that are based on IP packet traces. These methods are, in general, very accurate and are actively being used (e.g., by ISPs) but are only employed on wired networks. Furthermore, the popular DPI method is computational intensive and does not respect the privacy requirements of modern users or applications. In contrast, efforts have been made recently in the area of CR, often based on deep learning and in particular using a CNN architecture. While most approaches target the recognition of wireless technologies and modulation schemes, recent work has directed attention towards identifying specific traffic patterns as well. However, these works target only specific and very distinct types of traffic (e.g., differentiating YouTube from WhatsApp) and more generic traffic classes or transport protocols have not yet been considered. Overall, we can conclude that applying traffic recognition directly at the level of the wireless spectrum is still a completely open research challenge.

3 ORCHESTRA: seamless multi-technology management

"It's his M.O., isn't it? I mean, what are we, a team? No, no, no. We're a chemical mixture that makes chaos. We're... we're a time-bomb"

–Bruce Banner/The hulk (The Avengers, 2012)

The contributions presented in this chapter are based on the publications titled *"ORCHESTRA: Enabling Inter-Technology Network Management in Heterogeneous Wireless Networks"* and *"ORCHESTRA: Supercharging Wireless Backhaul Networks through Multi-technology Management"*.

3.1 Introduction

In Chapter 1, we introduced the heterogeneous and ever-growing nature of the wireless networks of today and tomorrow. Examples of technologies found across these diverse networks are Wi-Fi, LTE, Bluetooth, ZigBee, and satellite communication. These networks are typically managed in a static manner and each technology operates completely independent from one another. As such, leading to uneven load distributions among different technologies and infrastructure devices, suboptimal and inefficient use of wireless resources, performance degradations, and potential connection losses. Introducing the required coordinated management to these wireless environments is highly challenging, as each technology has unique capabilities and serves specific use cases

To address these problems, several solutions have been proposed at different layers of the network stack. The most notable solutions are MPTCP, the IEEE 1905.1 standard, LTE-LWA, and SDN-based solutions. Arguably, the most popular solution nowadays is MPTCP, capable of load balancing TCP flows across multiple network interfaces [75, 76]. However, MPTCP lacks intelligence and works only between two endpoints. IEEE 1905.1 is a data link layer solution, which allows dynamic flow redirection through an abstract MAC layer [45]. However, the downsides of IEEE 1905.1 are twofold. First, it only provides flow-level control, while packet-level control is needed for, among others, fine-grained load balancing that is fully utilizing all wireless capacity. For instance, consider the following hypothetical scenario: two links each have a capacity of 10 Mbps, while there are three traffic flows each having a rate of 6 Mbps. In this case, flow-level management is insufficient, while packet-level control could divide one of the flows across both links to have in total 9 Mbps of traffic on both links. Second, it was designed for specific network technologies (Ethernet, Wi-Fi, powerline, and MoCA) and does not support mobile networks. In contrast, a solution for mobile networks exists in the form of LTE-LWA that extends tradition LTE networks by offloading traffic over a Wi-Fi connection using tunnels [60,61]. Furthermore, also SDN-based solutions have been proposed that introduce a flow and device level of control over wireless devices and typically focus on seamless mobility [26]. In general, existing solutions lack intelligence, fine-grained control, and full independence towards communication technologies, protocols, and application domains.

To this extent, we introduce the ORCHESTRA framework that uses SDN principles to enable seamless multi-technology management of devices in heterogeneous networks. The framework consists of two components: (i) a fully transparent VMAC layer that unifies the various communication technologies underneath, offering a single point of connection to the upper layers. (ii) a centralized controller that is capable of managing both VMAC-enabled and legacy devices across the entire network based on received real-time monitoring information. On top of this controller, algorithms can be executed to optimize the overall network state and to increase network performance and user experience. In particular, the framework allows for the enabling of features such as seamless intra- and inter-technology handovers, packet-level load balancing and duplication of critical data, and dynamic path reconfiguration.

The remainder of this chapter is structured as follows: first, we introduce the ORCHESTRA framework in Section 3.2, discussing all the different buildings blocks and features in detail. Next, we describe how the framework can be used with different underlying communication technologies, such as Wi-Fi and LTE, in Section 3.3. We follow up by stating a number of relevant use cases where the framework can be adopted relatively fast in Section 3.4 and the presentation of a real-life prototype implementation in Section 3.5. Finally, we conclude with the results and a discussion in Section 3.6, where we provide an evaluation of the performance of the ORCHESTRA solution in comparison to MPTCP.

3.2 Framework architecture

The goal of the proposed framework is to offer a single solution to manage all different technologies within a network, regardless of the technologies and the type and scope of the network. The ORCHESTRA framework consists of two main parts: first, the transparent VMAC layer that manages physical interfaces on a device without modifying the underlying layers. It can be deployed on any device, both end-user or being part of the network (e.g., an edge or core node), and introduces seamless interactions between the different technologies. Second, following the SDN principle, we introduce the ORCHESTRA controller that has a global view of the network. The main responsibility of the controller is to manage the different VMACs across the network, based on real-time monitoring information. Both components are extensively discussed in the next subsections, where we highlight, among others, the different building blocks, features, and interactions with legacy (i.e., non-ORCHESTRA) devices.

3.2.1 Virtual MAC layer

In order to provide continuous and reliable connectivity, a key feature is the enabling of inter-technology handovers and roaming. As identified in Chapter 1, the current structure of the OSI network model obstructs this behavior. All communication technologies or standards operate completely independent from each other, as they each define their own lower layers of the network stack (in particular the MAC and physical layers). This means that connectivity is currently handled on an interface basis as each interface has its uniquely assigned network address and applications tend to bind on a single, specific interface. Consequently, changing between interfaces results in connection loss. We solve this issue by introducing a virtual MAC layer (VMAC) and abstracting connectivity from the user and applications. It also enables the implementation of functionality (e.g., load balancing) that works across multiple technologies. The general architecture and capabilities of the VMAC are shown in Figure 3.1.

The novel layer is placed above the existing data link layer and below the network layer, appearing transparent to both of them. The main responsibility of the VMAC is to forward incoming packets from the network layer to one (or multiple) of the underlying interfaces (i.e., technologies) under its control, or vice versa, forward packets received from the data link layer to the above network layer. Existing layers are thus not modified, do not require knowledge of the presence of the VMAC, and packets are still regularly passing through them. As such, abstraction and encapsulation, key principles of the OSI stack and the Internet, are maintained.

One of the main advantages is that there is only one interface (i.e., the VMAC) visible to the network and upper layers, while the underlying technologies, and their respective layers, are hidden away. This also means that also only a single IP address per device is needed, without requiring any additional overhead. In contrast to the IEEE 1905.1 standard, no unique virtual MAC address is required.

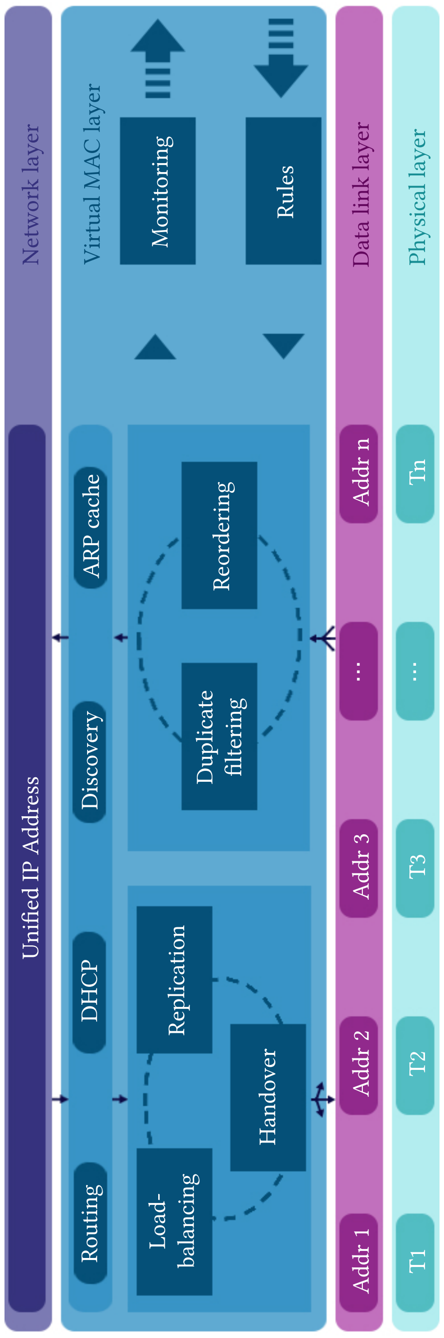


Figure 3.1: Overview of the VMAC layer with its position in the OSI model, its buildings blocks, and its offered functionality.

As such, the interaction with existing standards and protocols does not need to be altered. Furthermore, as the VMAC is capable of managing all the different interfaces, and these interfaces can be connected to different networks, it also needs to be able to route packets between different networks. For instance, between an edge network, that might be wireless, and a core network, that might be wired. Therefore, the VMAC incorporates the required bridging functionality as well.

Because of the single interface to the upper layers, and the abstraction of and control over all the underlying technologies, the VMAC captures all traffic and can therefore seamlessly handover between technologies. In particular, the VMAC introduces the following advanced functionalities:

1. Seamless handovers within a single technology between wireless endpoints or between different technologies to support mobility and leverage the best QoS available for a node across different technologies.

2. Packet-based load-balancing (and reordering at the other end) between two or more technologies to maximize network utilization.

3. Duplicating (and deduplicating at the other end) individual packets across several technologies to support high reliability.

The features stated above are enabled by the introduction of packet matching rules, to which incoming packets are matched. Additionally, statistics are gathered and forwarded to the central controller, while in turn, rules and commands are (ideally) received from the controller. This interaction is in more detail discussed in Section 3.2.2.1. Based on these rules (e.g., send all traffic to a specific node over a single interface) and commands (e.g., to perform a handover), the VMAC decides which interface handles the received packet and let the lower MAC layer take care of the actual transmission. With this granularity, the virtual layer can support packet-level control instead of flow-based handling, which allows for more versatility and control (cf. the hypothetical example in Section 3.1). This is in strong contrast to existing solutions such as IEEE 1905.1 and LTE-LWA.

Considering the general packet-flow, the VMAC introduces only minimal differences: on a sending node, when a packet arrives from the upper network layer, it enters the VMAC instead of directly going to the MAC of one of the underlying interfaces. Depending on the rule, that matches the header information of the packet, the VMAC decides to hand the packet over to the correct underlying interface, or in the case of duplication, multiple interfaces. On the receiving side, the incoming traffic is pushed from the data link layer interfaces to the VMAC, instead of directly being passed to the upper layers. In the specific cases of duplication and load balancing, some additional processing has to be done on the VMAC, before passing the packet upwards. We discuss these intermediate steps in Section 3.2.1.2.

The VMAC can be installed on any device, both consumer (i.e., endpoint) or infrastructure side, and applied in all kinds of networks (e.g., LANs, RANs, backhaul networks). However, we mentioned earlier that the VMAC requires bridging

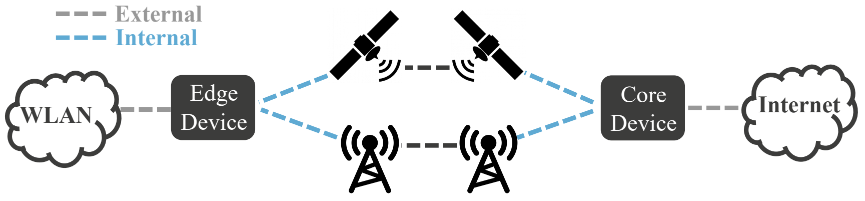


Figure 3.2: Illustration of internal and external interfaces in a backhaul scenario.

functionalities in the sense that it is capable of forwarding packets from, for instance, a wireless backhaul or edge network to a (wired) core network and vice versa. Note that this bridging functionality is typically required in ISP or backhauling use cases. In this case, we need to make a distinction between the interfaces, controlled by the VMAC, according to their functionality. We discriminate two types of interfaces: internal and external interfaces. The internal interfaces are part of the wireless backhaul network, meaning that they handle the traffic to and from edge nodes. In many cases, this might appear as its own subnet without direct access to an outside network. On the other hand, there is at least one external interface, which is part of the core network or an external network. This interface has outside connectivity and needs to handle packets from a different subnet and translate the IP addresses to the internal interface and vice versa. The VMAC is also responsible for handling the routing between the different subnets to ensure connectivity. The difference between internal and external interfaces is illustrated in Figure 3.2. Note that this functionality is not required on all nodes, for instance, not for endpoints at the edge of a network. A modified implementation (more lightweight) can be provided for such devices, in particular in the context of resource-constrained devices.

Subsequently, we first describe the basic building blocks that are necessary for the VMAC architecture. Afterwards, we explain in detail the features that the virtual layer has to offer.

3.2.1.1 Building blocks

Unified IP: In order to have a stable connection on the transport layer, it is vital that the IP addresses of the endpoints do not change. For this purpose, the VMAC only uses a single IP address for all interfaces. This IP address is (arbitrary) requested by the VMAC through one of the interfaces under its control. In the case of a handover, the IP address remains the same, while only the physical interface changes. As a consequence, the VMAC has to take care of Dynamic Host Configuration Protocol (DHCP) and relieve higher layers of it, otherwise the operating system gets into conflict with the network configuration as the same IP cannot be present on multiple interfaces. Furthermore, this also gives the controller an important role in actually managing all the different technologies and the VMAC in informing the controller correctly about what technologies it controls.

Dealing with multiple interfaces: When there are multiple interfaces active at the same time, for example when load balancing a traffic flow across multiple technologies, the normal Address Resolution Protocol (ARP) table of the operating system is not sufficient anymore. An operating system only matches an IP address to one of its interfaces, but if multiple interfaces are active (under the same IP address), the operating system would continuously overwrite the entry. Therefore, to cope with multiple simultaneously active interfaces and legacy devices, the VMAC needs to take care of the ARP handling. As such, keeping track of which IP address is reachable over which interface and if (potentially) an IP address is reachable over multiple interfaces. For this purpose, the VMAC maintains its own ARP cache and, upon receiving an ARP, stores the MAC address of the interface on which it received the ARP reply. The VMAC signals on all interfaces that it is available and remembers which IP to MAC tuple is available on which interface. To cope with the fact that an IP address can be reachable over more than one interface, a separate ARP cache is maintained per interface. This allows having multiple active connections at the same time, without experiencing problems regarding routing and discovering the other endpoint. When the VMAC receives an ARP request on a specific interface, the VMAC issues the transmission of an ARP reply only on that particular interface. This way a VMAC-enabled device can still communicate with legacy devices, both on the client or infrastructure side.

Monitoring: There is a continuous stream of configuration and monitoring information from the VMAC to the controller, allowing the controller to have a detailed and global view over the network. The configuration information includes what interfaces are available on the device and what their properties and capabilities are. An example of such an interface might be an LTE connection with a bit rate of 150 Mbps. Furthermore, also the state of a specific interface, if it is up or not, can be shared. In addition, the monitoring information includes statistics about these interfaces and the traffic going through them. This includes, for instance, the received packets or bytes per second, QoS information, recorded signal strength values for wireless links, and latency information. This monitoring information is sent to the controller using simple UDP packets.

Rules: The behavior of the VMAC is defined in the form of rules, typically set by the controller. On one hand, there are configuration instructions that specify the frequency of the transmission of monitoring reports to the controller. On the other hand, there are the rules that define how incoming packets (from both data link and network layers) should be handled. The latter includes the use of advanced functionalities such as load balancing or duplication. These rules consist of two parts and can be, conceptually, compared to OpenFlow rules. The first part states which packets should match the rule. This can, for instance, be done using source and destination IP addresses, port numbers, transport protocol types, and/or sequence numbers. The second part of the rule defines how the matching packets should be processed. This includes, for instance, simple forwarding over a single interface, load-balancing, or duplicating over multiple interfaces. For example, it is possible to directly forward all packets arriving within a specific IP range to one interface, while we balance a video stream across two or more interfaces to increase its throughput.

Furthermore, rules are sent to the VMAC by UDP packets. While the ORCHESTRA controller can handle this by a standard (UDP) socket, the VMAC checks the packet headers of incoming packets from the data link layer for the IP address of the controller and then extracts the required data from these packets. Note that the VMAC can also work without a controller present in the network and decide on its own transmission rules if necessary. It is also possible that the VMAC (or a local application running on top) decides to update the packet matching rules itself, for instance, in case of a disruptive network change (e.g., an interface going down). This is in order to minimize the impact on network traffic. Afterwards, the controller can, if needed, update these rules again, based on the received monitoring information and the global view, to have an optimal configuration across the entire network. This can, to some extent, be compared to the split-MAC principle employed by SDN controllers such as Odin [26, 54].

Discovery: When a device equipped with a VMAC joins a certain network, it needs to discover available controllers. Therefore, it broadcasts a (UDP) discovery message, to which the controller (or the most suited in case of multiple distributed controllers) responds. While the VMAC is not associated with a controller, it does not yet know the IP address of the controller. Consequently, the above-described procedure of receiving the controller's instructions based on the IP address is not yet possible. This can be solved by using a unique identifier in the UDP discovery message. Afterwards, the VMAC parses every incoming UDP packet, until a packet is received that is marked with the identifier at the beginning of the payload (first 64 bytes), and the IP address of the controller is learned.

3.2.1.2 Features

Handovers: A handover is an act of moving from one connection endpoint, like an AP or base station, to another connection endpoint. This can be done both within a single technology (referred to as intra-technology or vertical handover) or across different technologies (referred to as inter-technology or horizontal handover). Furthermore, different devices can perform a handover, for instance, a client device in a LAN or cellular network and an edge node in a (wireless) backhaul network. The decision for a handover is typically made centrally by the controller and it informs the respective VMAC about it. In the case of an inter-technology handover, only a single interface is currently used for the transmission of data and this interface is considered to be the active interface. The goal of the handover is to transfer the status of the active interface to another interface. Consequently, the packet matching rules are updated to reflect the change and to, instead, use the new active interface. In the case of an intra-technology handover, the active interface stays the same, but its endpoint is changing.

The following procedure is executed: first, the VMAC buffers outgoing packets on the active interface (for a brief moment of time), before either switching the endpoint of the active interface or changing the active interface to the new one. In case the active interface is changed, it sends out a gratuitous ARP to announce that the IP address is now associated with the MAC of the new interface and all relevant devices can update their routing tables. If the interface is not connected yet, the VMAC takes care of connecting it (e.g., performing an association procedure) and then switching to it. As a fail-safe, if it is not possible to set up a connection, the VMAC switches back to the old connection. When the handover is successful, the VMAC releases the buffer and starts transmitting again. Additionally, without instructed by the controller, in case of a link failure, the VMAC can decide to switch interfaces or connections by itself to maintain a connection on a certain technology.

To efficiently hand over a device, make sure all traffic streams are correctly delivered, and minimize the overhead, a protocol and synchronization steps need to be in place. This is done by exchanging several messages between all involved parties (e.g., the client device, the APs with the corresponding technologies, and, if necessary, switches) and agreeing on a time to perform the actual handover between technologies. In the described process we assume that a handover takes place between a client and the two different APs, but the procedure is identical for a handover between any other types of devices. As mentioned above, the process starts when the controller informs the different VMACs that a handover is imminent. All VMACs acknowledge this and one of them (typically the station or endpoint) starts a synchronization timer, which is also communicated to the other devices involved, as such agreeing on the current time. Next, each of the three devices announces the time window δ, needed to perform the actual handover. First, all parties agree (by exchanging ACKs) on the largest time δ among all actors. While, afterwards, in a similar fashion they also agree on a time t to initiate the handover. Finally, the handover is executed and afterwards the connection is tested. In case, the handover fails, both nodes fall back to the previous configurations. During the time δ of the handover, the VMACs on both the client and the new AP buffer the packets and transmit them after the handover has been completed and acknowledged.

In case the AP, in the previously described setup, is not equipped with a VMAC, a handover is still possible if the AP is managed by a SDN/Network Function Virtualization (NFV) controller, as such offering a form of legacy support. Otherwise, if the AP is operating fully independently, a VMAC-enabled device can still perform a handover and buffer its packets to lower the overall packet loss. However, no guarantees can be given on the overall duration of the handover and the overall performance, as this completely depends on the configuration of the APs in question.

Load-balancing: In contrast to a handover, while performing load-balancing, multiple interfaces are active and the network traffic is distributed according to a certain scheme to each interface. Opposite to most existing approaches, the VMAC introduces load-balancing at a packet level. For this, a simple weighted Round Robin load-balancing is used where a fixed number of packets are assigned to a specific interface before moving on to another interface. This can be done without introducing overhead as the VMAC only needs to forward the packet to another interface that is active. However, if packets that are part of a continuous data flow are being sent across different interfaces, no guarantees can be made on the order of arrival at the other endpoint, especially in a wireless context (e.g., due to different latencies across different technologies or external interference). Therefore, at the other end, packets need to be reordered in the VMAC before being passed to the upper layer. Otherwise, transport layer protocols, especially TCP, will react in an unpredictable way. This reordering is done on a flow basis, per TCP session, and according to the transport layer protocol header, more specifically, the port number and the sequence number. Additionally, the source and destination IP addresses are used to assign different packets to a flow. The VMAC keeps track of the sequence numbers and buffers packets when a sequence number is missing. If the out-of-order packet arrives, it will be forwarded immediately, while the previously buffered packets are forwarded afterwards in the correct order, until a next out-of-order packet is identified. As some packets may never arrive at the receiving end, a timeout for missing packets is provided. This timeout depends on the rate of the traffic flow as this determines the turnaround of sequence numbers, used to identify out-of-order packets. Furthermore, the timeout is dynamically adjusted by monitoring the throughput and kept as low as possible to minimize negative effects on the transport layer protocol. When the timeout is reached, all packets that are available are forwarded in an ordered fashion to the network layer.

Duplication: Duplication is a useful method to achieve high reliability as it strongly increases the probability of a packet to arrive at the other endpoint. At the sending side, the VMAC enables this by copying an incoming packet and transmitting it across different interfaces (depending on the specified rule). However, at the receiving side, the VMAC cannot simply push the receiving packets to the network layers, as the same packet can potentially be forwarded multiple times. In turn, this can trigger unwanted behavior on the application layer or a reaction of the transport protocol (especially in the case of TCP). As such, the receiving VMAC is responsible for filtering out these duplicates (i.e., performing deduplication). This is done in a similar manner to the reordering of packets upon performing the packet-level load-balancing, as described above. The VMAC maintains a hash map of packets that it already received. The packets are identified by source and destination IP address, transport layer protocol, IP identifier and IP fragmentation offset. A timeout is in place to prevent memory over usage. This timeout is, similar to the one used for missing packets, depending on the actual flow rate and therefore monitoring is necessary to adjust it appropriately. As the maximum

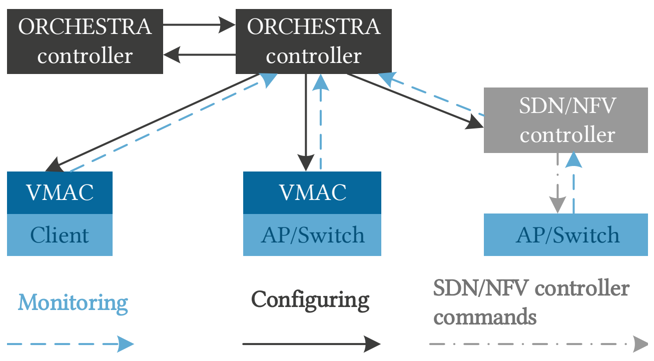


Figure 3.3: The controller architecture and its communication.

number of duplicates that can arrive is known, the entry can be deleted as soon as this number is reached. Otherwise, the entry is deleted after the timeout. Note that none of the existing methods discussed in Chapter 2 allows for such a fine-grained form of duplication.

3.2.2 Controller

While the VMAC allows for fine-grained MAC-level control inside individual devices, the ORCHESTRA controller is the heart of the proposed framework and enables multi-technology management and orchestration across the entire network. Following the SDN principle, the controller takes control from the individual devices and their respective VMACs. The controller keeps track of all connected VMACs and issues instructions and updates to all of them to optimally configure the entire network. Furthermore, the controller is also capable of communicating with (other) SDN controllers, as well as individual infrastructure devices such as APs and switches. This is done by operating at a new and higher hierarchical layer, above of existing SDN controllers. An overview of the architecture can be seen in Figure 3.3. The architecture allows for more network control and a single central point to implement management logic. This contrasts with existing solutions like MPTCP and LTE-LWA. The communication with existing SDN controllers or infrastructure entities, allows for legacy support and an easier roll-out of the ORCHESTRA solution. This in opposition to, for instance, the IEEE 1905.1 standard, which requires more disruptive network changes and was never widely adopted. Additionally, the controller can also be distributed to increase scalability and reliability. In the next subsections, we elaborate more on the details of the controller, in particular on the communication aspect and the offered management possibilities.

3.2.2.1 Communication and interfacing

Here, we discuss all the interactions that are possible between the ORCHESTRA controller and the different entities in the network.

VMAC: The communication between the controller and the VMAC is lightweight and was already introduced before. In particular, the discovery of the VMAC and the two-way communication between the VMAC and the controller to, respectively, transmit monitoring information and rules, are discussed in Section 3.2.1.1.

Other SDN controllers: As it is unlikely that all devices within the managed network are immediately equipped with the novel VMAC, it is important to support legacy devices. In many networks, SDN controllers are already in place that offer certain management functions that can be exploited for devices not using the virtual layer. Examples of such frameworks, like ODIN and 5G-EmPOWER, are discussed in Section 2.2.3 [54, 55]. Interfacing with these SDN controllers requires more effort than the lightweight communication with the VMAC, as they usually do not support a built-in so-called northbound interface that is accessible through external communication. However, most controllers (e.g., the Ryu OpenFlow and the 5G-EmPOWER controller) offer application support insofar as you can write an application on top of the controller that interfaces with the controller and implements some higher-level functionality. This can be exploited by creating a northbound interface running, as such an application on top of the controller. The application handles the communication and translation of information from the SDN controller to the ORCHESTRA controller as well as commands from the ORCHESTRA controller to the SDN controller. The ORCHESTRA controller typically enforces station handovers (identified by MAC addresses) towards wireless SDN controllers (e.g., 5G-EmPOWER). While towards wired SDN controller (i.e., OpenFlow controller) the focus lays on traffic flow management and routing (e.g., adding flows, deleting flows, changing output ports). Vice versa, all SDN controllers provide the ORCHESTRA controller with the information that is available within the framework. For instance, traffic information (e.g., source and destination addresses, port numbers, or throughput), device information (e.g., MAC addresses or capabilities), and network conditions (e.g., link capacities or signal strengths). As such, we allow the ORCHESTRA controller to have an overview of, and to optimize, the various networks or segments managed by different controllers. Note that the exact communication can be realized through different kinds of communication frameworks or protocols, for instance, using the lightweight and performant ZeroMQ framework [136].

Infrastructure devices: However, not all devices in the networks of today are managed by SDN controllers. While client devices cannot be managed at all without the presence of a VMAC or a SDN controller, infrastructure devices, such as APs, can in most cases be controlled through a variety of standardized protocols. This typically depends on the specific type of device. For instance, continuing the popularity of the SDN paradigm, OpenFlow is prevalent as the communication protocol towards switches. This means that switches can be controlled directly by utilizing the OpenFlow protocol and send flow-based rules. For APs this is less straightforward, but they often support configuration through the Network Configuration Protocol (NETCONF) with Yang as the modeling language. As such, a Yang model can be developed for every type of device, according to the exact capabilities of that device. Finally, in theory, it is also possible to extend the supported protocols, but this would require updates to the devices as well. If the option exists to update the (endpoint) devices, it might be better to move directly to either an SDN solution or the installation of the proposed VMAC.

Distributed ORCHESTRA controllers: To ensure scalability and reliability, the ORCHESTRA controller can be distributed. The communication between these different distributed ORCHESTRA controllers is handled in a similar fashion as the communication with other SDN controllers. The controller maintains an eastbound interface that includes the discovery of other controllers through broadcasting, as well as the transmission of heartbeats to maintain the connection. Only relevant information to other controllers is exchanged to reduce network traffic. This information includes common devices that are in the range of multiple controllers, especially if a device might be moved from one controller to another. Information is exchanged either by request or by informing another controller that one of the nodes is leaving the control of the current controller. For instance, consider a device that is in the range of two APs and is connected to one of them. One of the APs is in the region of the first controller, while the other AP is managed by a second controller. As both controllers have information on the device, the state is shared among both controllers. If a handover is needed, because of a newly computed assignment would place it in the region of another controller, the controller currently responsible for the device, informs the other controller to take over the device. The new controller, in turn, updates its flow rules and AP configuration and acknowledges the handover. Afterwards, the old controller deletes the remaining flow rules and the AP configuration and only further monitors the device. Note that this entire exchange happens fully transparent to the moved device and its VMAC. Finally, as the communication between different ORCHESTRA controllers is similar to the interactions with other SDN controllers, the same underlying communication frameworks and protocols can be used.

3.2.2.2 A global view in one location

The ORCHESTRA controller has two other components besides the communication interfaces. The first part consists of a data store where all received information is aggregated and combined into one state model, representing the whole network under consideration. This includes information about the VMACs (e.g., throughput, RSSI, latency), about infrastructure devices (e.g., how many clients are connected, the capabilities, and performance), and about the SDN controllers (e.g., the local view of that controller). All of this information is stored in a single format in a large store or database that can be shared among the potential several controllers.

Second, the controller also offers a northbound interface which applications, running on top of the controller, can use. This allows for implementing decision-making logic and algorithms on a single location in the network, managing different devices and network technologies in a ubiquitous manner. As network technologies are abstracted and the controller takes care of the abstraction layer, this greatly simplifies the implementation of such management logic applications. In Chapters 4 and 5, we introduce such algorithms that optimize the network-wide throughput. Other algorithms can, for instance, also focus on Time-Division Multiple Access (TDMA)-based scheduling or even on energy efficiency. The intelligence schemes use the aggregated information of the storage as input to provide a certain configuration for the network. For instance, in the case of the algorithms in Chapters 4 and 5, a device to connection point to technology mapping is created, as well as routes for all traffic flows. Based on this configuration, the necessary commands are issued to the corresponding devices across the network to actually roll-out the particular configuration.

3.3 Applicability to different wireless technologies

In this section, we discuss how underlying communication technologies can be used in conjunction with the ORCHESTRA framework. We focus mainly on IEEE 802 and 3GPP technologies, and highlight, in the case of LTE, potential challenges that can be encountered when integrating the technologies.

3.3.1 IEEE 802

The IEEE 802 standards define a physical layer and a MAC layer for different technologies, such as Ethernet (IEEE 802.2), Wi-Fi (IEEE 802.11) or wireless Personal Area Networks (PANs) such as ZigBee (IEEE 802.15). These two layers define the physical transmission over the medium and how the medium should be accessed. For instance, in the case of Wi-Fi the MAC defines a Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) scheme. The IEEE 802 standards do not define any layer higher than the MAC layer, which means any network layer communication can be used. By default, this is IP, as it is the prevalent network protocol. Integrating IEEE 802 technologies into the VMAC is therefore straightforward and can be seen as plug and play. As the VMAC is positioned on top of the existing MAC layers of the technologies, it simply receives the incoming packets from the MAC layer and similarly, injects the outgoing packets into the appropriate MAC layer(s). No modifications to the underlying technologies are necessary, as intended.

3.3.2 LTE

Similarly, to the IEEE technologies, 3GPP technologies are defined by specifying a physical layer and a MAC layer. The transmission over the medium is defined by the physical layer, while the access to the medium is defined by the MAC layer, offering the possibility to use Frequency Division Duplex (FDD) or Time Division Duplex (TDD) modes. Contrary to IEEE 802 technologies, 3GPP technologies split the control and management plane from the data plane, comparable to the key principle of the SDN paradigm. For this reason, 3GPP specifies a set of entities responsible to provide authentication and connectivity to the UEs. The following procedure is followed: when a UE tries to connect to a network, it first talks with the eNB, which notifies the Evolved Packet Core (EPC) to authenticate the user subscription. If a valid subscription is found, the eNB establishes a GPRS Tunneling Protocol (GTP) tunnel to the gateway to grant the UE access to the network of the operator. If there is no valid subscription, the eNB cannot simply create a tunnel to the gateway and therefore, the client does not get connectivity with the network. A series of different interfaces are defined between the management entities and the access to external networks still utilizes GTP tunnels as a means of transportation.

As such, we can say that LTE by default carries legacy functionality in the form of these GTP tunnels. While GTP tunnels might have an advantage in managing clients in a traditional sense and in providing a secure channel across another technology than LTE (e.g., in LTE-LWA), it has the downside that all (data) traffic flows through the gateway, the endpoint of the tunnel. When considering use cases that provide services that are close to the edge of the network (i.e., close to the user device), this is a major disadvantage as you are producing additional traffic in the core network. Furthermore, in the scope of the proposed VMAC this also introduces limitations as, among others, packet-based load balancing and duplication becomes infeasible. This is due to the fact that the GTP tunnels do not allow to detect individual traffic flows, and it becomes infeasible to aggregate data flows that originate from another technology and network. As such, the standard LTE core architecture is not compatible with the VMAC. This can be addressed by including an additional header in the packet with flow information, but this would create additional overhead. However, alternative solutions are available.

3.3.2.1 LTE-LWA

As introduced in Section 2.2.4, the 3GPP community introduced the cooperation of LTE and Wi-Fi technologies in order to offload traffic from the cellular networks [60, 61]. In particular, LTE-LWA was introduced in 3GPP Release 13 [62, 64]. As shown in Figure 3.4, LTE-LWA allows for both a co-located and a non-co-located deployment of the two technologies. In the first case, shown at the left of Figure 3.4, the Wi-Fi AP and LTE eNB are connected through an external interface, denoted as Xw. On the other hand, the physical integration of the AP in the eNB is also possible, as shown at the right of Figure 3.4. In both cases, the aggregation of user plane data flows, transmitted over the two different technologies, occurs in the Packet Data Convergence Protocol (PDCP) layer. In turn, the LTE-WLAN Aggregation Adaptation Protocol (LWAAP) is responsible for encapsulating the data packets to tunnel them over the Wi-Fi connection.

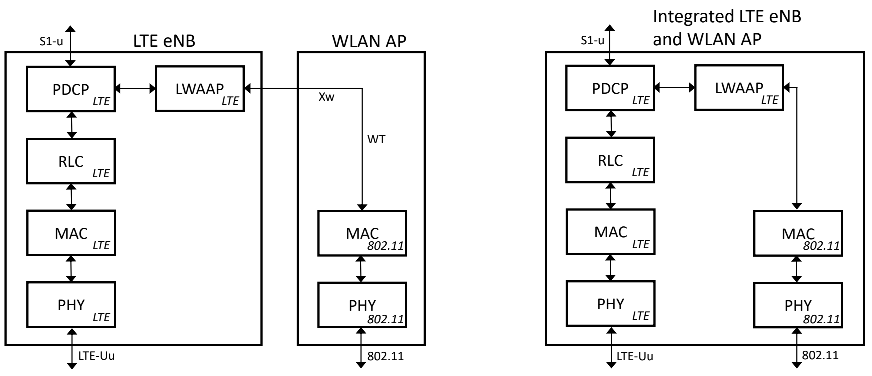


Figure 3.4: LTE-LWA user plane architecture with external Wi-Fi AP (left) and internal Wi-Fi AP (right) [62].

This LTE-LWA architecture provides an aggregation point for the LTE and Wi-Fi technologies, before traffic flows disappear in the GTP tunnels. As the VMAC is intended to bind over different interfaces, offering a single upwards connection, the aforementioned architecture can also be used for the installation of the VMAC. The VMAC can replace (or be merged with) the PDCP layer, offering additional features like packet-level load balancing and duplication. Furthermore, the deployment of the VMAC architecture removes the need for the tunnel over the Wi-Fi connection, as the VMAC is fully IP-based. As this adapted architecture requires changes to the current standards, it counteracts our initial idea of transparency to upper and lower layers, while potentially limiting the adoptability of the presented approach. Note that also only a select number of LTE-LWA deployments is currently planned world-wide, as mentioned in Section 2.2.4.

3.3.2.2 MEC architecture and Local Breakout

To allow for more flexibility and control over network resources, shorter routes, and the introduction of an IP interface, it was proposed to break open the above-mentioned GTP tunnels for data traffic [137]. This idea originates from the desire of telecommunication operators to have more insight in, and control over, the data traffic [138]. Furthermore, it is also proposed to enable edge computing, more efficient access to resources and services for clients (e.g., for gaming), and for 5G connectivity in VANETs [139–141]. Within the context of edge computing, the Multi-access Edge Computing (MEC) architecture has been developed, as shown in Figure 3.5. The essential part is located in the base station, where user IP data packets can be intercepted by decapsulating GTP packets. Those IP packets are rerouted to the edge network (i.e., MEC server/gateway) by the introduction of a breakout rule that changes the path. This mechanism, known as Local Breakout (LBO), was standardized in 3GPP Release 15 [142, 143]. Note that it does not affect the management part of LTE that still uses the standard GTP tunnels. The

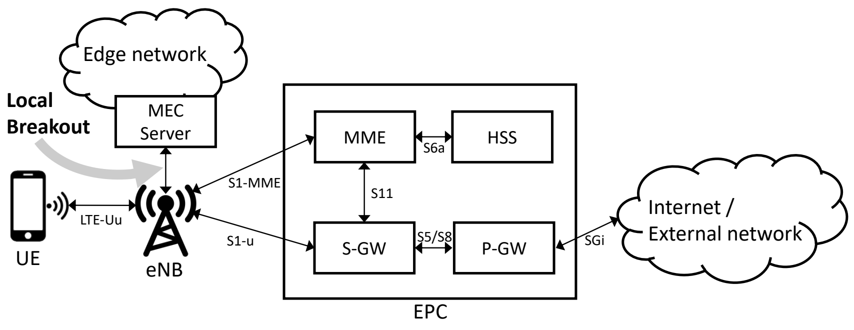


Figure 3.5: Basic MEC architecture

main advantage of this architecture is that the local traffic can be offloaded from the RAN to reduce the end-to-end latency of edge services and save core network load. The MEC system was developed independently from the already existing LTE networks. However, it is currently being considered in the further development of the 5G technology, since edge computing has been marked as one of the key elements required to enable future IoT services [144].

The MEC architecture, and in particular the LBO, opens opportunities for the use of the VMAC layer in an 3GPP context. First of all, it is possible to integrate the presented VMAC layer with the MEC architecture by installment on the MEC server. This enables the use of the ORCHESTRA features, like seamless handovers or load balancing, over the LTE connection and the other present communication technologies (e.g., Wi-Fi) in the edge network. Furthermore, in a non-edge computing context, we can still use the LBO to intercept the IP packets from the connected UEs that are ORCHESTRA-enabled. These IP packets are then routed to whichever VMAC layer is installed at the infrastructure side. Ideally, the VMAC is positioned close to the edge of the network. As such, flows that are split across different routes can be merged as early as possible, limiting the differences in, for instance, latency and arrival time, for the split flows due to the different link conditions. The VMAC can, for instance, be installed on an additional device connected to the eNB, similar to the MEC server. From this device, the merged flows can be routed again to the core network (if required) in order to reach the Internet or external networks. The VMAC layer can also be installed in the core network itself or on intermediate nodes between the eNB and the EPC. This all depends on the network architecture of, for instance, the telecommunication operator. Essential is that the VMAC is positioned in such a manner that split flows going over different routers (i.e., technologies) can be routed to it. Note that using the LBO technique and routing the IP packets directly, removes the need and overhead of the GTP tunnels (for data traffic). In Section 3.5, we present such a prototype implementation that utilizes the LBO mechanism to allow for an ORCHESTRA setup with Wi-Fi and LTE technologies. Furthermore, future work should study the placement of the VMAC layer in the RAN in more detail. The latter is discussed in Section 7.3.1.

3.4 Use cases

Two of the most straightforward use cases for the deployment of the proposed ORCHESTRA framework are LANs and RANs. These environments are occupied by a significant number of different communication technologies that can be used simultaneously. This can benefit the modern services and applications that produce various traffic streams and have stringent quality requirements, as described in Chapter 1. In this section, we discuss other use cases as well, to demonstrate the versatility and applicability of the ORCHESTRA framework. For each use case we clarify the advantages for end-users (e.g., better services) and the gains for the network operators (e.g., additional chargeable services or easier network management). Furthermore, in contrast to the LANs and RANs scenarios, these additional use cases require no modifications on the consumer devices, as such providing a more straightforward deployment.

3.4.1 Enhanced satellite networking solutions

As still two-thirds of humankind has no access to wired or wireless Internet, interest has grown in satellite networks with global coverage capabilities [145, 146]. Furthermore, satellite technologies are also being used to provide Internet access (i.e., Wi-Fi) on board of ships. Initially, Geosynchronous (GEO) satellites were used to provide connectivity to a large area, at the cost of a very low data rate and high latency because of the long distance to the satellites. Therefore, a hierarchical spot-beam architecture has been proposed where a GEO satellite controls a group of Low Earth Orbit (LEO) satellites that each offer connectivity to a smaller area on the ground [146]. However, because of the use of the mobility of LEO satellites, a much more dynamic environment is created, which requires advanced SDN solutions to manage the frequent horizontal handovers between satellites [146]. This is where ORCHESTRA comes into the picture as it can manage the handovers in a more transparent way, thereby reducing the management burden for the satellite network operator. Because of the fact that recalibrating and positioning the satellite receiver to a new satellite (i.e., a handover) takes time, dual-receiver solutions have been proposed, where a second receiver is directed to another satellite, while the first one remains connected with the old satellite. In this case, ORCHESTRA can provide a smooth handover and manage both interfaces to the receivers.

As satellite networks, by nature, introduce a relatively large delay and connectivity issues can occur, the cooperation with other technologies brings clear advantages. For instance, a ship that travels near a coastline can be in range of land-based LTE networks, which often offer better QoS than a satellite link. The implementation of ORCHESTRA in the ship's receiver (i.e., the edge node) allows for the simultaneous use of both LTE and satellite networks. This results, among others, in a more stable and performant Wi-Fi network on board of the ship for the crew and passengers.

3.4.2 Enabling autonomous driving

The vehicles on our roads are becoming more intelligent and will, eventually, become fully autonomous. An essential aspect of this evolution is the communication between these vehicles and (road-side) infrastructure and between vehicles mutually. This communication is required to support features like platooning, provide updates on the condition of the road and traffic ahead, or even optimal lane usage. Currently, two main concurrent technologies have been developed: IEEE 802.11p (the base for the IEEE 1609 and European ITS-G5 standard) and LTE-V [14, 147]. As both technologies will be deployed, for instance, alongside our roads and in our cars, load balancing can be used to off-load traffic and devices across the two technologies. This can help, among others, to keep latency low and allow for high-speed communication. Furthermore, the duplication of critical data can be used to offer more reliable communications. Finally, note that ORCHESTRA can also be considered for the in-car network, as these autonomous vehicles will also typically provide Internet connectivity in the car for there passengers.

3.4.3 Edge computing for large IoT deployments

Edge computing is the paradigm where intelligence and computational resources are (partially) moved away from the traditional cloud environment to the edge of network [148]. As such, it allows addressing concerns like response time, battery life constraints, bandwidth efficiency, and data safety or privacy [148]. Edge computing has been identified as one of the key enablers of the large-scale adoption of the IoT paradigm [144, 148]. For this reason, it is also a critical aspect of the 5G technology roadmap and research [138, 144]. At the edge, large numbers of interconnected devices (e.g., sensors, cameras, intelligent displays, end-user devices, ...) will be present, while different communication technologies will be used. In this heterogeneous environment, ORCHESTRA can aid by offering inter-technology network management to, among others, enable more efficient communication to reduce energy consumption and offload traffic streams to support large volumes of data and users. An interesting direction for future work is the application of ORCHESTRA in the MEC architecture, as discussed in Section 3.3.2.2.

3.4.4 Extended coverage in rural areas

While a majority of people live in hyper-connected cities, there is still a significant amount of people that live in more rural areas, for instance, in the southern part of Belgium. These houses often have an old DSL line, originally for telephone communications, that is used for Internet access. However, the limited capacity of these lines is not sufficient to meet the growing demands of end-users. As houses are sparsely distributed with large distances between them, it is also too expensive for telecommunication companies to deploy high-speed broadband solutions. Therefore, recently, hybrid-DSL with LTE solutions have been proposed where the home gateway is capable of receiving both [85]. This is also known

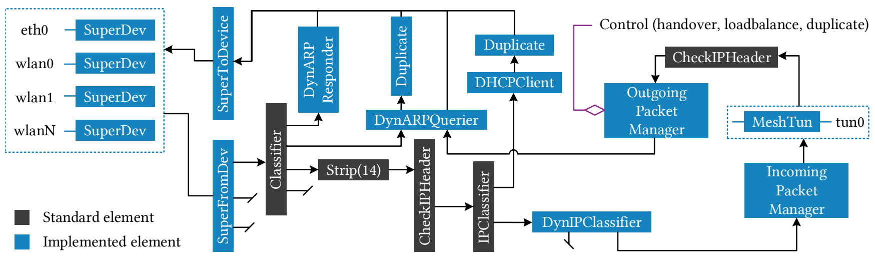


Figure 3.6: The implementation graph for Click showing the different elements used.

as DSL-LTE bonding. Traffic is divided among both links, thereby increasing the available capacity. Often, MPTCP is deployed at both endpoints to utilize both interfaces. However, each MPTCP connection needs to be created at one end and split again at the other end, which raises the management burden. Furthermore, the network operator needs to manage all the different MPTCP connections going to all end-users. The deployment of the ORCHESTRA framework heavily reduces the complexity of the management as it transparently handles all interfaces and traffic flows without the need for merging and splitting flows. Moreover, it also supports other traffic types than TCP. Note that we compare the performance of the ORCHESTRA solution to MPTCP in Section 3.6.

3.4.5 Wireless community networks

Because of the high availability, low-cost, and ease-of-deployment of wireless LAN equipment, wireless community networks have emerged [149]. In these wireless communities, broadband connectivity and a number of free services (e.g., free community-wide Voice over IP (VoIP)) is offered by a dense deployment of APs connected in a wireless mesh with fixed wireless access. These wireless community networks are traditionally connected to the Internet through a mobile network (e.g., 3G or 4G) and/or one or more Wi-Fi point-to-point links, possibly over a long distance. Note that a nearly identical use case can also be found on large events or festivals, where a wireless mesh is deployed to provide connectivity for visitors or services, while a wireless backhaul network is installed [150]. In both cases, the ORCHESTRA framework can be introduced to manage the wireless backhaul network. This enables features such as transparent handovers and load balancing between the different paths and technologies while reducing the deployment and management effort. Moreover, ORCHESTRA can also be used to manage the wireless community network itself.

3.5 Prototype Implementation

The current implementation of the prototype uses the Click modular router on a Ubuntu 16.04 machine [151]. We opted for Click as it allows for fast and high-level prototyping, which is handy for ongoing research. This is in contrast to a kernel-level implementation for a more finalized framework. While we use existing Click elements for basic packet handling, we implemented the VMAC logic in new elements to support the proposed functionality. The basic packet flow is shown in Figure 3.6. Multiple interfaces are connected to the *SuperFromDevice* and *SuperToDevice* block which take, respectively, care of forwarding packets from and to interfaces. Below we discuss the packet flows for both incoming and outgoing traffic.

3.5.1 Incoming traffic

For incoming packets, the header is stripped and the class of the packet is detected. This is done in the elements *Classifier*, *Strip (14)*, *CheckIPHeader*, *IPClassifier*, and *DynIPClassifier* in Figure 3.6 (denoted in grey). As the VMAC takes care of the generation of ARP requests and replies, it needs to filter out ARP at this point. A received ARP reply indicates that the virtual layer did send out a request because a packet in the buffer is waiting to be transmitted. If the arrived packet is an ARP request however, the virtual layer immediately replies (*DynARP responder*). Furthermore, as the VMAC does not provide a DHCP server or similar, it has to forward DHCP requests to the interface that is connected to the corresponding network. However, the VMAC has a DHCP client of its own that takes care of requesting IP addresses (*DHCPClient*). This is necessary as the VMAC uses only one IP address for all interfaces. Note that we do not implement in this prototype the different internal and external interfaces, as discussed in Section 3.2.1.

If the incoming packet is determined as data traffic, the next step is the *IncomingPacketsManager* which implements the logic of the proposed features (e.g., deduplication or reordering). As it is incoming traffic, packets need to be reordered or deduplicated if load balancing or duplication is used. The VMAC also checks for controller traffic at this point and consumes the packet if this is the case, in order to change its configuration or rules. Afterwards, the data packet is forwarded to the *Tun* interface and made available to higher layers.

3.5.2 Outgoing traffic

In the other way around, outgoing traffic is handled in a similar manner. After identifying the packet, there are two main components. First, the *OutgoingPacketsManager* which implements load balancing, duplication, and the handover logic. This component decides to which interface a packet is forwarded. Furthermore, the commands regarding which rules should be used are managed by this component as well. Second, there is the *DynARPQuerier*. Here, ARP requests are generated and transmitted across multiple interfaces, while an outgoing packet is buffered if no ARP entry exists for the requested IP. As soon as a reply arrives or if the entry exists, the packet is forwarded to the underlying interface, which takes care of the actual transmission.

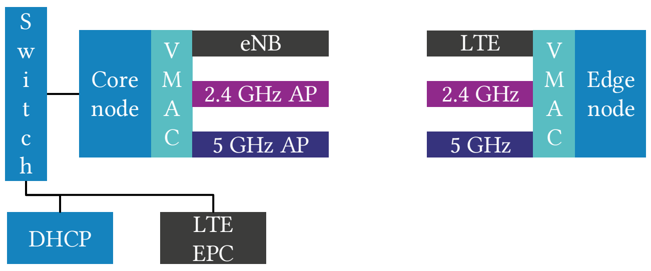


Figure 3.7: The setup of the prototype including all devices.

3.6 Evaluation and discussion

In this section, we compare the capabilities of the ORCHESTRA framework to MPTCP for the three described key functionalities. First, we start with a description of the evaluation setup. Next, we show the results for handovers between two interfaces, followed by load balancing across two interfaces, and ending with duplication across two interfaces. Each interface is using a different technology, namely Wi-Fi and LTE.

3.6.1 Experimental setup

The prototype setup consists of several components, displayed in Figure 3.7. The prototype represents the setup for the deployment of the VMAC on the devices in a wireless backhaul scenario. The core components that are equipped with the VMAC are the following: (i) the edge node, which is close to the end-user and consists of a device that acts as an LTE UE and a Wi-Fi client. (ii) the core node, which is connected to the wired core network and connected over Ethernet to an LTE eNB and a Wi-Fi AP. Additionally, there is an EPC that manages the LTE network (e.g., authentication), and an external DHCP server. Both are connected via a switch to the core node. Note that the setup would be the same for deployment in LAN, except for the fact that the edge node would be replaced by a client device, and the core node would be called an infrastructure device.

The AP consists of an APU2c4 board using the LEDE operating system with an IEEE 802.11n Wi-Fi card using a 20 MHz channel [152]. Furthermore, it is configured through OpenWrt as a bridge between the wireless and wired network. The base station is installed on a computer with an Intel core i7 8700k processor and 16 GB of RAM with a USRP B210 Software-Defined Radio (SDR) using a 15 MHz channel. It uses a modified srsLTE implementation to create an eNB that allows to remove GTP tunnels. The eNB is managed by the OpenAirInterface EPC [153]. The edge device, the core device, and the EPC device are all Intel NUCs with a core i5 4250U processor and 16 GB of RAM. For the UE, a Huawei E3372 LTE USB stick is used, while the DHCP router is an arbitrary home router. Note that previous versions of the prototype also contained an Ethernet connection.

In the following scenarios, except stated otherwise, LTE and Wi-Fi (on the 5 GHz frequency band) are employed for the two interfaces. We evaluate our solution with both TCP and UDP streams, generated through *iperf*, while comparing it to MPTCP version 0.94 [154]. All tests are conducted in an office environment where there is a distance of 2 m between the edge node and both networks. Each scenario is repeated 10 times, and average results are reported.

3.6.2 Seamless and transparent multi-technology handovers

In this scenario, we consider that a handover between Wi-Fi and LTE (or vice versa) is initialized every 30 seconds, which is indicated by the vertical lines in the figures. Furthermore, we consider a 1 Mbps flow of traffic for 120 seconds. Figures 3.8 and 3.9 show, respectively, the results in terms of throughput and latency. MPTCP can handle handovers with a 1 Mbps stream to a limited extent in terms of throughput (it loses connection, but can reestablish it), as can be seen in Figure 3.8. However, latency increases heavily as MPTCP loses connection and first needs to establish a new sub-flow. In contrast, ORCHESTRA can seamlessly switch between technologies and maintains a constant throughput, while keeping the latency low (cf. Figure 3.9). In particular, we see that there is a downtime of 21 % for MPTCP, while this is 0 % for ORCHESTRA. The two handovers from Wi-Fi to LTE (at 30 s and 90 s) result in a connections loss of respectively 3 and 2 s. This corresponds to values reported in literature [77,78]. However, switching from LTE to Wi-Fi at 60 s, results in a connection loss of 30 s, as the connection is lost until switching back to Wi-Fi. The reason for this significantly higher connection loss is unclear, and is potentially caused by some misconfiguration. Finally, note that the seamless connectivity provided by ORCHESTRA is the case for both TCP and UDP streams. This is also in contrast to MPTCP which, by its nature, only supports the TCP protocol.

In order to demonstrate the backward compatibility of ORCHESTRA, an additional small experiment was performed where we tested three variations in the configuration of an AP and station. Note that we do not use the wireless backhaul scenario here, as a difference in configuration is more likely to occur in a LAN scenario. In the first configuration both the AP and station are ORCHESTRA-enabled devices with a VMAC. The second configuration consists of an ORCHESTRA-enabled AP as before, while the station is a legacy device (without a VMAC). The third considered configuration contains two legacy devices. For the legacy devices, we used two Intel NUCs with Ubuntu 16.04 installed. As Ubuntu machines are used, we had to simulate the handover of devices without a VMAC layer. By default, Ubuntu reacts very slow, if at all, when a connection is breaking down and multiple technologies are available. For a wireless connection, this can easily take 15 s or more, in which case the connection completely drops. The value of 15 s was experimentally determined. Note that this value for more end-user oriented OSs, like Windows or macOS, will typically be lower. Therefore, to have *iperf* not break down, a script monitored the link continuously and if no traffic was detected for four seconds, it switched the route to the correct interface. This

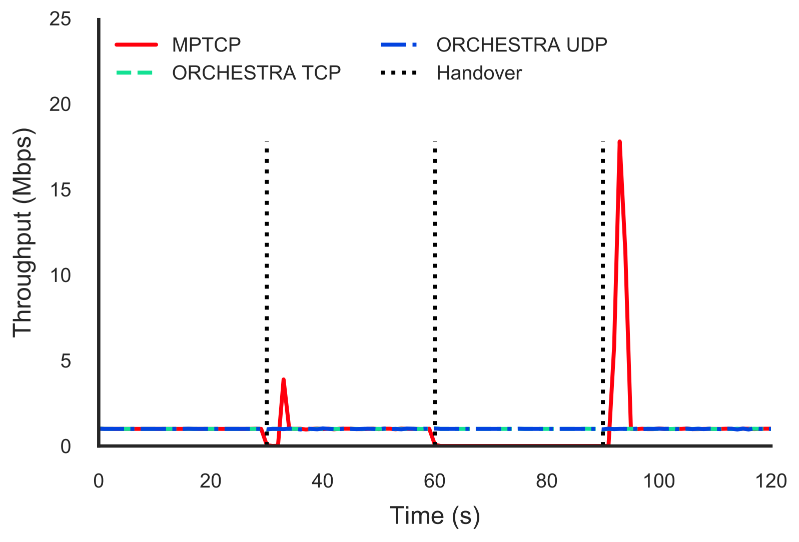


Figure 3.8: Handover performance of MPTCP and ORCHESTRA in terms of throughput.

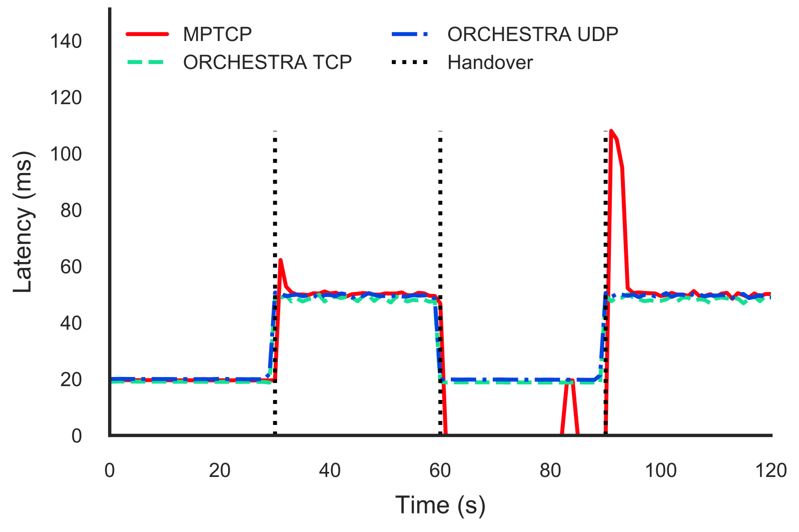


Figure 3.9: Handover performance of MPTCP and ORCHESTRA in terms of latency.

approach can, to some extent, be compared to band steering where an AP forces a station to another frequency, except for the monitoring script. We show throughput results for both a TCP and UDP traffic flow of 6 Mbps.

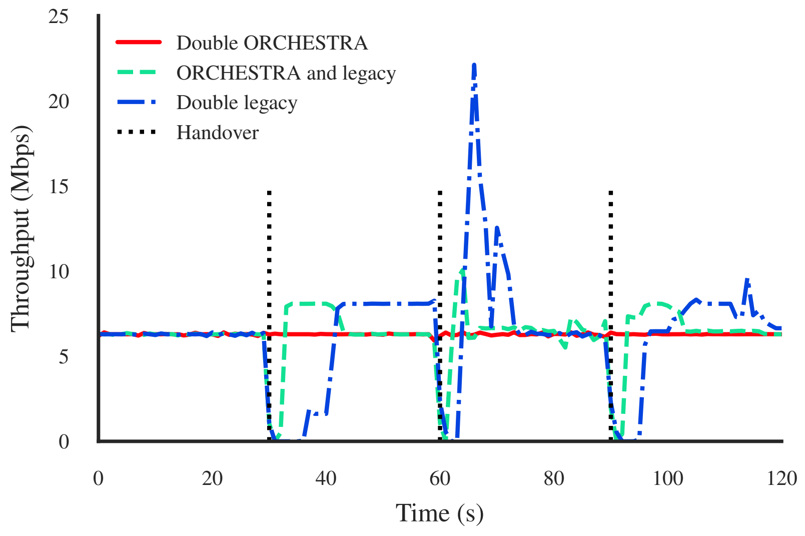


Figure 3.10: Comparison of handover performance for ORCHESTRA and legacy devices for TCP traffic.

From Figures 3.10 and 3.11 it is clear that all three variations result in different throughput patterns and that connection drops occur under the presence of legacy devices. For the scenario with two legacy devices, we see that after performing a handover the traffic drops completely because the underlying connection was lost as there is no coordination between the two devices. On the other hand, *iperf* with TCP tries to overcompensate by increasing the amount of traffic until on average a throughput of 6 Mbps is reached, as soon as the connection is reestablished. This can clearly be seen in Figure 3.10, as the throughput heavily increases and reaches up to 23 Mbps for Wi-Fi and 8 Mbps for LTE. This means that the application has to handle the connection loss and as soon as it detects it, it needs to reestablish the connection, causing a significant downtime. Note that not all applications can cope with this behavior. UDP traffic exhibits a similar behavior, as shown in Figure 3.11. However, it is more resilient to sudden link failure as it does not require ACKs and packets are sent regardless if a connection exists or not.

In the second case, consisting of one VMAC-enabled and one legacy device, different behavior is experienced as the VMAC, can detect much faster than the OS if a connection is dropping. Upon a connection loss, it can easily switch to another technology and send packets over the new connection. This can be seen in Figures 3.10 and 3.11 as the drop in throughput is not as long as with the two legacy devices. While improving the downtime, the drop itself is not completely avoidable as the handover is done without informing the other device. This is still in stark contrast to the third scenario with two ORCHESTRA devices

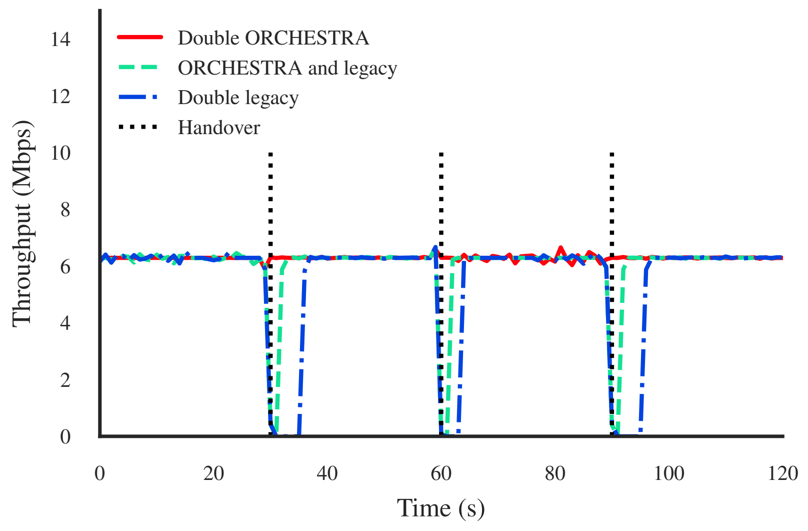


Figure 3.11: Comparison of handover performance for ORCHESTRA and legacy devices for UDP traffic.

where drops are completely mitigated and seamless handovers are performed. Figure 3.10 shows that TCP itself is not reacting at all to the handovers and that the throughput remains constant throughout the run of the experiment. This heavily improves performance and the traffic flow (i.e., the underlying TCP protocol) does not need to overcompensate for the time the connection is down. The responsibility to take care of the network connectivity is removed from the application and is completely in control of the network intelligence or the network operator, who have a better overview of the network.

3.6.3 Fine-grained packet-level load balancing

In order to demonstrate the packet-level load balancing capabilities of ORCHESTRA using our prototype, the two interfaces are actively used at the same time. Both for a TCP and UDP stream we configure the VMACs on the two devices to balance the traffic evenly (i.e., according to a 50/50 distribution) across both the Wi-Fi and LTE interface. We compare this to MPTCP that is configured to use the default (round-robin) RTT scheduler. This scheduler sends a fixed number of packets over a specific interface, before rotating to the next interface. For this experiment, we once again use a traffic flow of 6 Mbps.

The results in terms of throughput and latency are shown in, respectively, Figures 3.12 and 3.13. On average the desired rate of 6 Mbps is achieved by both MPTCP and ORCHESTRA. The latter does this for both TCP and UDP traf-

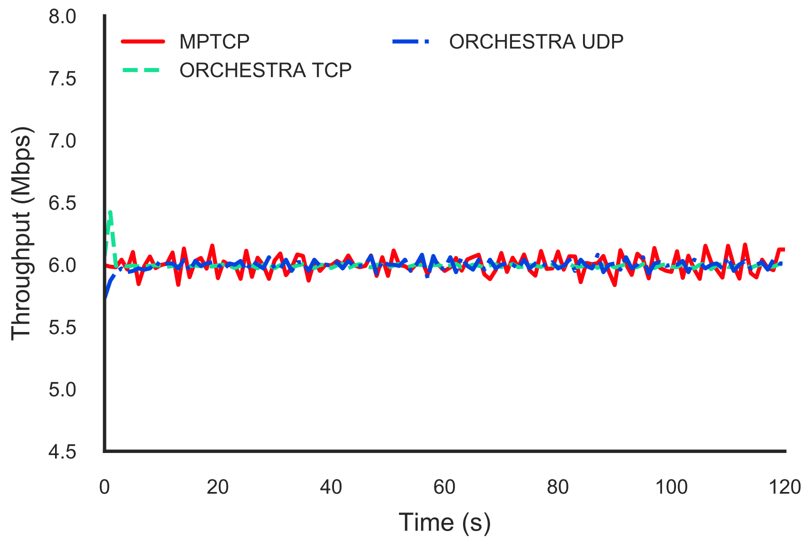


Figure 3.12: Load balancing performance of MPTCP and ORCHESTRA in terms of throughput.

fic flows. Figure 3.13 indicates that there is a significant increase in latency (of 40.6 %) for the TCP stream when using ORCHESTRA. We clearly see that latency builds up for the first 20 to 25 s before stabilizing. While, in contrast, the UDP flow and the TCP flow with MPTCP experience lower amounts of latency throughout the experiment.

The explanation for the behavior experienced with TCP when using the ORCHESTRA framework is twofold: First of all, the challenge in load balancing the individual packets of a traffic flow across two different wireless technologies lays in the different latency properties of those technologies. This potentially results in out-of-order packet arrivals. MPTCP circumvents this problem as its scheduler sends out a fixed number of packets after each other on the same interface before using the other one. This results in a 67 %/33 % distribution of packets across both interfaces in favor of Wi-Fi. As the Wi-Fi connection has lower latency, this partially explains the difference in latency with ORCHESTRA that really balances all packets evenly in a 50 %/50 % distribution. Furthermore, with the MPTCP protocol, large amounts of packets are sequentially sent on the same interface. This series of packets will thus always arrive in order at the receiver side (under circumstances with no packet-loss, as is the case here). However, note that the throughput of the TCP flow also slightly fluctuates, due to differences in latency when the scheduler switches between the interfaces.

Second, ORCHESTRA uses a reordering mechanism at the receiving VMAC to cope with the potentially out-of-order packet arrivals (as explained in Sec-

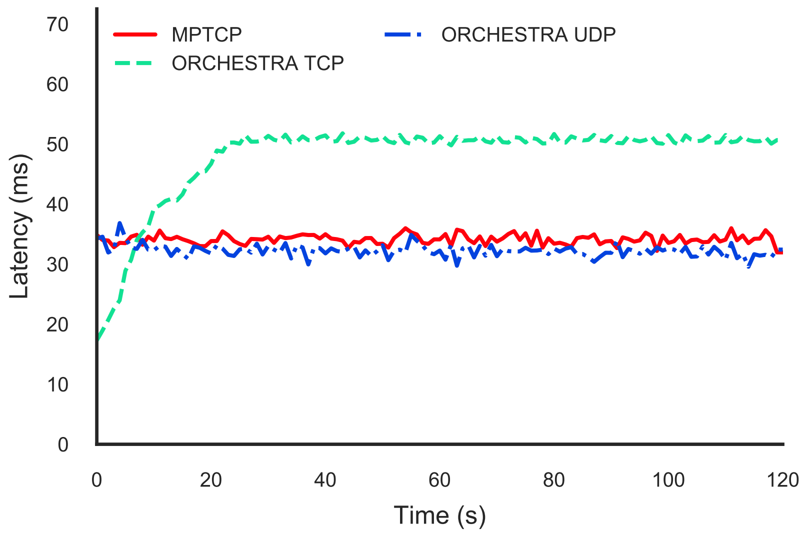


Figure 3.13: Load balancing performance of MPTCP and ORCHESTRA in terms of latency.

tion 3.2.1.2). For TCP, this reordering is necessary because no assumptions can be made about the capabilities of the upper layer. In this case, out-of-order packets are placed in a hash map, until missing packets have been received (or a timeout is triggered). The packets are reordered according to TCP sequence numbers before being delivered to the upper layers. Since packets are distributed across both technologies according to a perfect 50/50 scheme, frequent reordering is needed, causing the increase of latency before stabilizing. The smaller fluctuations throughout the remainder of the experiment are caused by the TCP rate control mechanisms that react on the slightly varying inter-packet times. Note that the UDP flow does not experience this behavior, due to the lack of rate control algorithms.

In order to further demonstrate the functionality of the packet-based load balancing, and to investigate the impact of the reordering, the following experiment is conducted: similarly to before, we transmit both a 6 Mbps TCP and UDP flow that is split across two interfaces of the prototype, namely the 5 Ghz Wi-Fi interface, and the LTE connection. However, in contrast to the previous experiment we do not only balance the load evenly but vary the percentages: at the start the traffic stream of 6 Mbps is balanced 50/50 % across both available interfaces. After 30 s this is altered into 30 % of traffic over Wi-Fi and 70 % over LTE. At the 60 s mark we return to the initial 50/50 % configuration, before ending up with a 70/30 % for, respectively, Wi-Fi and LTE.

From the results for both TCP and UDP, shown in Figures 3.14 and 3.16, it is clear that the packet-based load balancing works as intended, as the traffic is in

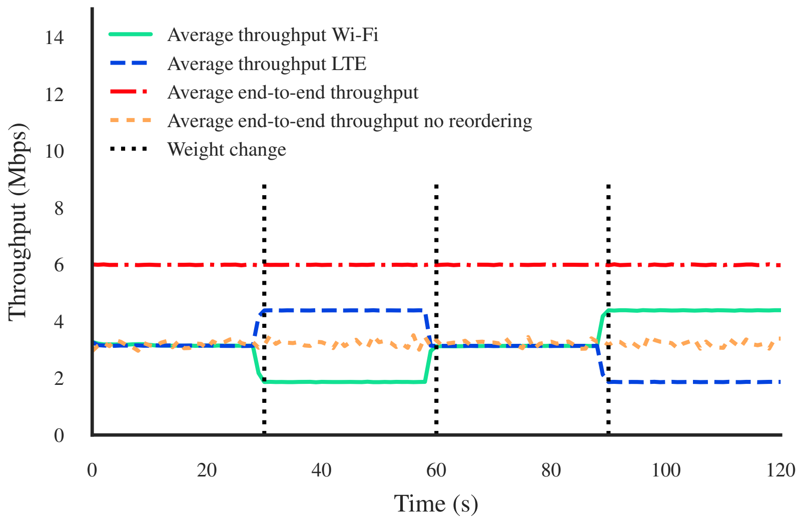


Figure 3.14: 6 Mbps TCP flow load-balanced over two technologies with a weight change from 50/50 (Wi-Fi/LTE) to 30/70 to 50/50 to 70/30.

both cased distributed across both interfaces according to the set weights. Both the TCP and UDP flows achieve a stable throughput of 6 Mbps. The need for packet reordering is clearly shown in Figure 3.14 where the TCP flow only achieves a throughput of 3.2 Mbps when reordering is disabled. Furthermore, Figure 3.15 shows the observed latency across the entire length of the experiment with TCP traffic. Similar to the previous experiment, latency increases for the first 25 s of the experiment. Afterwards, latency varies depending on how the packets of the flow are scheduled across the two technologies.

Overall, we can say that the increased throughput and flexibility of the packet-based load balancing with reordering, comes at the cost of a slightly increased latency (in comparison to MPTCP). This should be addressed in future work to try to close the gap in latency between ORCHESTRA and MPTCP in this scenario. However, we clearly demonstrate that the technology abstraction of the VMAC layer is working as intended and that, even with TCP, we can precisely (on a packet-level) load balance a flow (both TCP and UDP) among multiple technologies with different characteristics. This MAC-level scheduling can be extended in future work to include packet scheduling across different competing technologies to minimize interference.

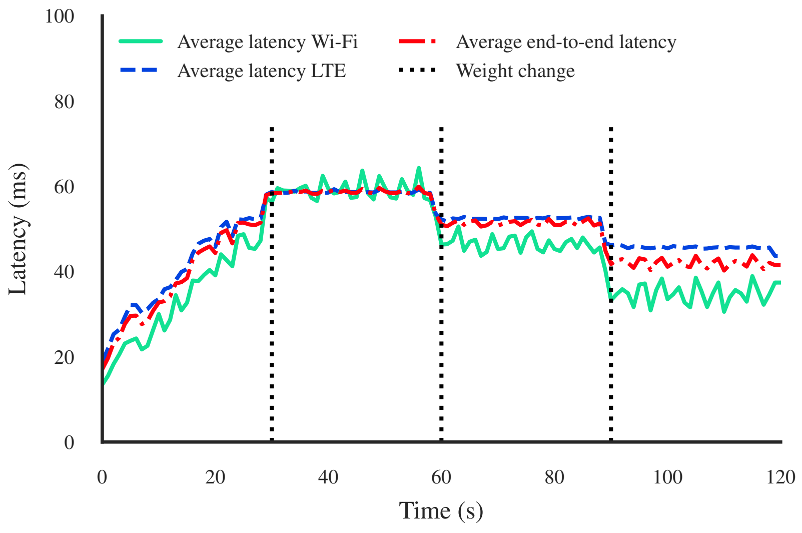


Figure 3.15: 6 Mbps TCP flow load-balanced over two technologies with a weight change from 50/50 (Wi-Fi/LTE) to 30/70 to 50/50 to 70/30.

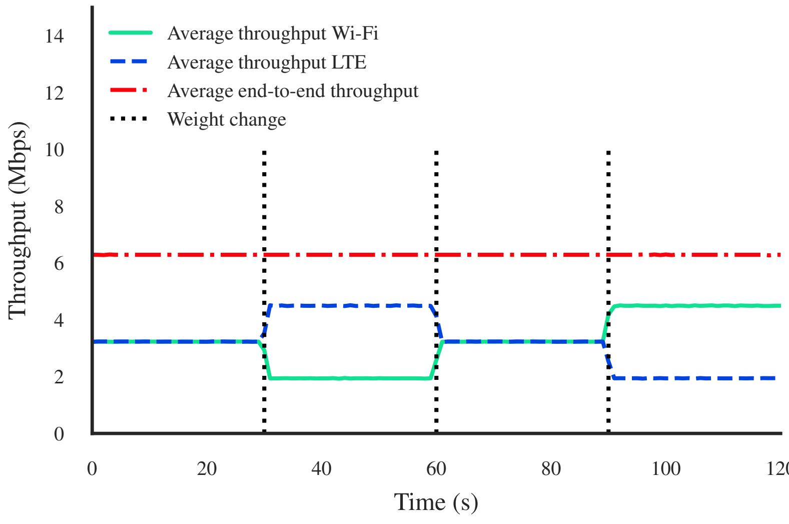


Figure 3.16: 6 Mbps UDP flow load-balanced over two technologies with a weight change from 50/50 (2.4 GHz/5 GHz) to 30/70 to 50/50 to 70/30.

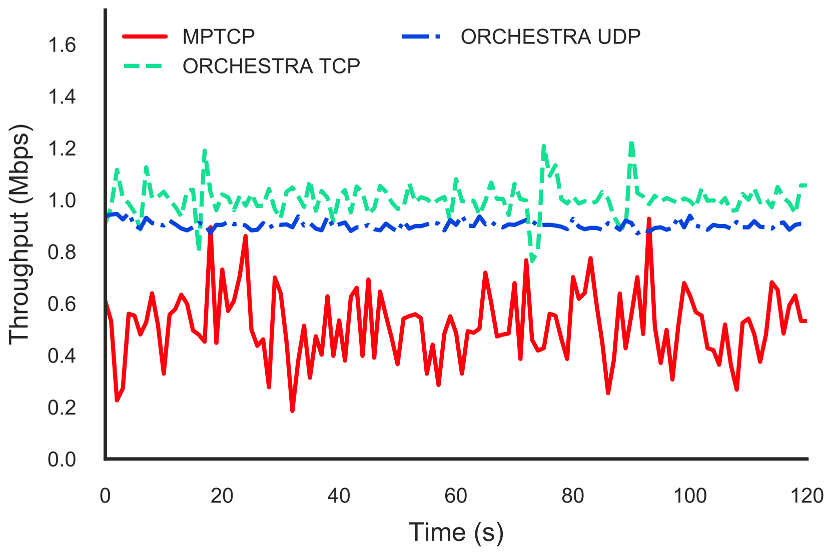


Figure 3.17: Duplication performance of MPTCP and ORCHESTRA for throughput.

3.6.4 Duplication of critical data in unreliable environments

For the duplication scenario, there are, similar to the load balancing scenario, two continuously active interfaces with LTE and Wi-Fi as their respective technologies. However, instead of balancing the traffic flow, each incoming packet is copied and sent out over both interfaces. We emulate an unreliable environment by dropping packets on each link, with a chance of 25 % per packet per link. A flow of 1 Mbps is used for both TCP and UDP traffic. In both cases, duplicates need to be detected and removed by the deduplication functionality in the VMAC. Below, we will explore the implications and performance of that, compared to MPTCP. For MPTCP the redundant scheduler is used instead of the default RTT scheduler [155]. This redundant scheduler sends the data replicated through all of the active subflows available, while back-up subflows are established to send retransmissions.

The results of this scenario are shown in Figures 3.17 and 3.18, respectively, for throughput and latency. For MPTCP we notice a large drop in throughput since instead of 1 Mbps, only around 0.5 Mbps is achieved. Furthermore, a corresponding increase in latency can be noted as well. In contrast, ORCHESTRA achieves the full 1 Mbps while the average latency stays below 20 ms as well. This is the case for both TCP and UDP traffic. ORCHESTRA achieves this by duplicating packets transparent to the transport protocol compared to MPTCP, which uses different TCP subflows that each suffers from packet loss. Moreover, the small fluctuations in TCP and UDP throughput (in the ORCHESTRA case) are due to the fact that some packets are still not reaching the receiver as the duplicates can be dropped

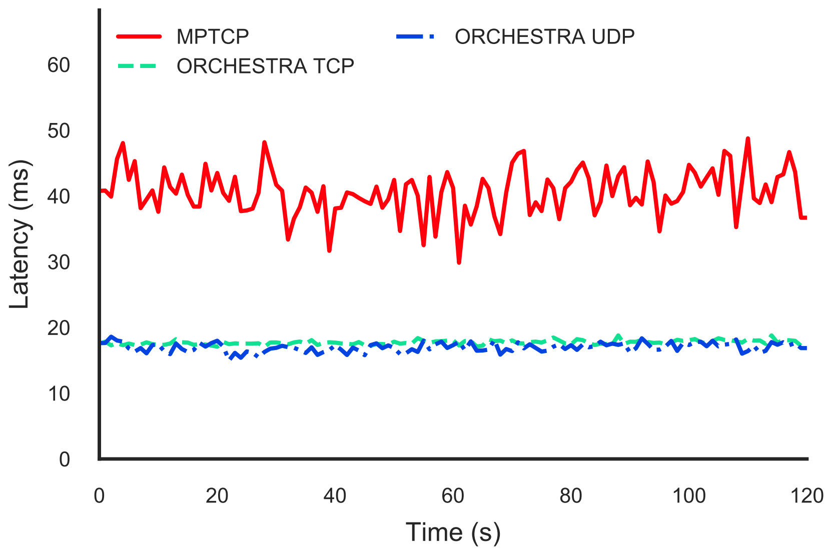


Figure 3.18: Duplication performance of MPTCP and ORCHESTRA in terms of latency.

on both links (with a chance of 12.5 %). However, it is clear that ORCHESTRA significantly increases redundancy, especially in unreliable environments.

3.7 Conclusion

This chapter presents the ORCHESTRA framework that enables seamless inter-technology network management. It consists of two main components: a VMAC layer and a centralized controller. The VMAC layer transparently bonds different physical interfaces into one connection towards the upper layers, while the central controller enables coordination and a global view over the entire network. The framework enables features like inter-technology handovers, packet-level load balancing, and duplication. Furthermore, we present the implementation of ORCHESTRA in a real-life prototype that supports the following communication technologies: Ethernet, Wi-Fi (in both the 2.4 Ghz and 5 Ghz), and LTE. Our evaluation shows that all features work as intended for both TCP and UDP traffic, as such outperforming the default industry solution MPTCP.

Disclaimer

The work presented in this chapter regarding the ORCHESTRA framework was equally contributed by Patrick Bosch, Ensar Zeljković, and myself.

4 Real-time flow management for heterogeneous networks with both wired and wireless connections

“Intelligence is the ability to adapt to change”

– Stephen Hawking (1942 - 2018)

The contributions presented in this chapter are based on the publications titled *”A transparent load balancing algorithm for heterogeneous local area networks”* and *”Flow Management and Load Balancing in Dynamic Heterogeneous LANs”*.

4.1 Introduction

The ever-growing numbers of consumer devices and communication technologies, together with the increasing volumes of traffic, have raised a management puzzle. On one hand, there are the stringent quality requirements of modern services (especially multimedia applications) that are sensitive to network disruptions and degradations. On the other hand, the static management of wireless networks and devices. Despite supporting multiple technologies, modern devices tend to connect to the Internet using a single technology, based on predefined priorities. For instance, they opt for Ethernet, before 5 GHz Wi-Fi, before 2.4 GHz Wi-Fi. In some cases, the user or OS can (manually) override these priorities, but no automatic methods exist for dynamically switching between interfaces or using multiple ones simultaneously.

In Chapter 3, we introduced the ORCHESTRA framework to enable seamless inter-technology management for all kinds of networks. This framework enables, in a coordinated manner, features like inter-technology handovers, flow rerouting, or packet-based load balancing. However, we did not yet define the intelligence that utilizes these features to optimize the network and its performance. Existing algorithms for multi-technology environments (i.e., LANs or RANs) focus mostly on the development of theoretical models that assume detailed knowledge of both flow throughput requirements and dynamic network conditions. Furthermore, the existing approaches are technology specific and not suited for coping with QoS requirements as they neglect the specific nature of wireless networks (e.g., interference, link quality variability) and the typical behavior of, for instance, TCP traffic flows.

In contrast, we introduce in this chapter a dynamic flow management approach that aims to increase network-wide throughput in networks that have both wired and wireless connections, for instance, LANs. We compare two different MILP formulations that both calculate the optimal path configuration for all traffic flows in the network. The first formulation does so by maximizing the throughput of the different individual traffic flows, taking into account TCP fairness mechanisms. Furthermore, we introduce the notion of collision groups as the encapsulation of a group of interfering links or stations, respectively, denoted as Link Collision Groups (LCGs) and Station Collision Groups (SCGs). The second formulation provides the optimal configuration to maximize the throughput across all the SCGs. In particular, we improve upon existing solutions in three ways. First, rather than assuming to know flow throughput requirements and dynamic network conditions, this is estimated using real-time monitoring information. Second, our approach takes into account the specific nature of wireless networks, where users do not have dedicated network resources but a shared medium instead. Third, our approach offers improvements in terms of scalability and computational efficiency, thereby allowing for more realistic use cases.

The remainder of this chapter is structured as follows. In Sections 4.2 and 4.3 we present the two mathematical programming formulations that, respectively, use a flow-based or collision group-based approach. Next, we discuss in Section 4.4 how the algorithms can be deployed on top of a network management solution (e.g., ORCHESTRA) and how parameters are determined. Finally, in Section 4.5, we round-up this chapter by providing an evaluation of the proposed approach, using a combination of thorough simulations and a prototype implementation.

4.2 Flow-Based Scheduling MILP Formulation

In this section, we present the first out of two MILP formulations that aim to optimize the network throughput. This algorithm targets heterogeneous environments with both wired (e.g., Ethernet) and wireless (e.g., Wi-Fi) connections. The formulation is denoted as Flow-Based Scheduling Formulation (FBSF) throughout the remainder of this chapter.

4.2.1 Network Model

The architecture of the network is modeled as a multi-graph defined as a quadruple (N, T, L, G^L) where:

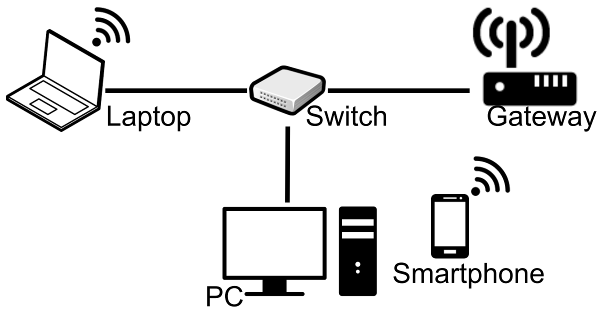
- N is the set of nodes $\{n_1, n_2, ..., n_n\}$. These nodes represent the different devices within the network.

- T is the set of technologies $\{t_1, t_2, ..., t_t\}$ and $c_t \in \mathbb{R}_{\geq 0}$ represents the capacity of $t \in T$. We define capacity as the total amount of data (typically measured in Mbps in this dissertation) that can be transmitted over the technology (or connection) in question, at a certain moment in time. This set T can, among others, consist of Ethernet and Wi-Fi. Note that two instances of the same technology with different characteristics are modeled as different technologies. For instance, an Ethernet cable of 10 Mbps and one of 100 Mbps are modeled separately. The same can be said about Wi-Fi in, respectively, the 2.4 and 5 GHz band.

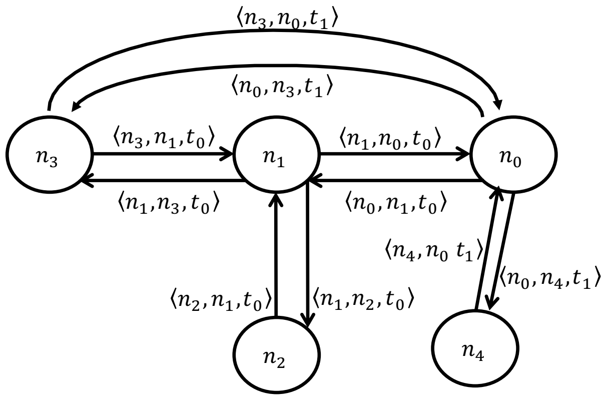
- L is the set of links. A link is characterized by a triple $< s_l, d_l, t_l >$ with $s_l \in N$ the source node, $d_l \in N$ the destination node and $t_l \in T$ the technology.

- G^L is the set of all Link Collision Groups (LCGs). An LCG $g \in G^L$ is defined as: $\{l_1, l_2, ..., l_i \in L \mid t_{l_1} = t_{l_2} = ... = t_{l_i} \wedge c_{t_{l_1}} = c_{t_{l_2}} = ... = c_{t_{l_i}}$ $\}$. In other words, an LCG encapsulates all the links that share the capacity of a technology and where the transmissions can interfere with each other. For instance, if two devices (or nodes) are connected by Wi-Fi with an AP, the links between the devices and the AP are in the same LCG, as their capacity is shared.

An example of a simple (LAN) topology is shown in Figure 4.1a. This network consists of a gateway (that also serves as a Wi-Fi AP), a switch and three devices, a PC, a laptop, and a smartphone. Both the PC and laptop are connected via a switch to the gateway, while there is also a Wi-Fi connection for the laptop and the smartphone. This network can be represented by means of the previously described model, as is shown in Figure 4.1b. Each of the devices is represented by a node $\{n_0, n_1, n_2, n_3, n_4\}$. There are also two different technologies $\{t_0, t_1\}$, representing the Ethernet cables (assuming that all cables have the same capacity) and the Wi-Fi connection. In an actual network representation there will likely also be a second Wi-Fi network (e.g., a 5 GHz connection in addition to the present 2.4 GHz connection), but this was omitted from this example for readability reasons. For each of the connections between the nodes, there are two links present in the model. One link per direction. For instance, the Ethernet connection between the gateway and the switch is described by the links $< n_0, n_1, t_0 >$ and $< n_1, n_0, t_0 >$. In this example there are seven LCGs: there is one LCG that consists of the four links corresponding with the Wi-Fi connection $\{< n_0, n_3, t_1 >$, $< n_3, n_0, t_1 >$, $< n_0, n_4, t_1 >$, and $< n_4, n_0, t_1 >\}$. All other links have their



(a) Network topology


$\langle n_3, n_0, t_1 \rangle$
$\langle n_0, n_3, t_1 \rangle$
$\langle n_3, n_1, t_0 \rangle$
$\langle n_1, n_0, t_0 \rangle$
$\langle n_1, n_3, t_0 \rangle$
$\langle n_0, n_1, t_0 \rangle$
$\langle n_4, n_0\ t_1 \rangle$
$\langle n_0, n_4, t_1 \rangle$
$\langle n_2, n_1, t_0 \rangle$
$\langle n_1, n_2, t_0 \rangle$
n_3
n_1
n_0
n_2
n_4

(b) Multi-graph representation

Figure 4.1: Example of a simple heterogeneous network topology together with its multi-graph network model representation.

own LCG, due to the fact that Ethernet is full-duplex so there are no collisions between the traffic in the two directions over the wired connection.

In addition to the network topology, traffic flows going through the network also need to be modeled. Let us define F as the set of all flows. A flow is a triple $< s_f, d_f, r_f >$ with $s_f \in N$ the source node, $d_f \in N$ the destination node and $r_f \in \mathbb{R}_{\geq 0}$ the desired rate of the flow. In our example, a flow of, for example, 1 Mbps from the gateway to the laptop is represented as the triple $< n_0, n_3, 1 >$.

4.2.2 MILP formulation

The flow scheduling problem under consideration is modeled as an MILP formulation, which consists of the necessary inputs, decision variables, an objective function, and a set of constraints. This model can be solved (i.e., to find the optimal value of the decision variables, given the constraints, and objective function) using a variety of optimization algorithms and heuristics, for instance, offered by some commercial solvers like Gurobi and CPLEX [156, 157].

The inputs for the algorithm consist of the previously described network and flow model, as well as the following sets:

- $S^{L_n} : \{\forall l \in L \mid s_l = n\}$; describes the set of all the links that have node $n \in N$ as source.
- $D^{L_n} : \{\forall l \in L \mid d_l = n\}$; describes the set of all the links that have node $n \in N$ as destination.
- $S^{F_n} : \{\forall l \in L \mid s_l = s_f\}$; describes the set of all the links that have the same source as flow $f \in F$.
- $D^{F_n} : \{\forall l \in L \mid d_l = d_f\}$; describes the set of all the links that have the same destination as flow $f \in F$.
- $X^{F_n} : \{\forall l \in L \mid d_l = s_f\}$; describes the set of all the links that have as destination the source of flow $f \in F$.
- $Y^{F_n} : \{\forall l \in L \mid s_l = d_f\}$; describes the set of all the links that have as source the destination of flow $f \in F$.

We define the following decision variables:

- $\lambda_{l,f} \in \{0, 1\}$; this variable represents the path for every flow. If a flow $f \in F$ passes over a link $l \in L$ then $\lambda_{l,f} = 1$, otherwise it equals 0.
- $\tau_f \in [0, r_f]$; this variable defines the assigned rate (bandwidth) of a flow $f \in F$. Note that the upper bound of the interval is defined by the desired rate of the flow (denoted as r_f).

As an objective function, the model maximizes the total assigned rate (bandwidth) over all flows:

- $max \sum_{f \in F} \tau_f$

Finally, we define the following constrains:

- The capacity constraint makes sure that for each LCG, the total capacity of the technology is not exceeded: $\forall g \in G^L : \sum_{l \in g} \sum_{f \in F} \lambda_{l,f} \cdot \tau_f \leqslant c_{t_g}$. Note that this constraint is not linear as it contains a multiplication of two decision variables, however it can be linearized by replacing the multiplication by a novel decision variable [158].
- The flow conservation constraints make sure that the links assigned to a flow form a loopless path (from the start to the destination node of the flow):
  - Every path has exactly 1 link departing from the start node of the flow: $\forall f \in F : \sum_{l \in S^{F_n}} \lambda_{l,f} = 1$

- Every path has exactly 1 link arriving at the destination node of the flow: $\forall f \in F : \sum_{l \in D^{F_n}} \lambda_{l,f} = 1$
- Every path cannot have a link arriving at the start node of the flow: $\forall f \in F : \sum_{l \in X^{F_n}} \lambda_{l,f} = 0$
- Every path cannot have a link departing from the destination node of the flow: $\forall f \in F : \sum_{l \in Y^{F_n}} \lambda_{l,f} = 0$
- For every node on the path of the flow, except the start and destination, there can be at most one link departing from that node: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in S^{L_n}} \lambda_{l,f} \leqslant 1$
- For every node on the path of the flow, except the start and destination, there can be at most one link arriving in every node of the path: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in D^{L_n}} \lambda_{l,f} \leqslant 1$
- The total number of links departing from each node should be equal to the total number of links arriving at that node, except for the start and destination nodes: $\forall f \in F, \forall n \in N \setminus \{s_f, d_f\} : \sum_{l \in S^{L_n}} \lambda_{l,f} = \sum_{l \in D^{L_n}} \lambda_{l,f}$
- Of all the links, in both directions, between two nodes $n, m \in N$, there can at most 1 link be part of each path: $\forall f \in F, \forall n, m \in N : \sum_{l \in S^{L_n} \cap S^{L_m}} \lambda_{l,f} \cdot \sum_{l \in S^{L_m} \cup S^{L_n}} \lambda_{l,f} = 0$

4.2.3 TCP fairness

Existing models and algorithms for load balancing assume that the rate τ_f of each flow can be chosen by the algorithm. Although this offers an extra degree of freedom, and therefore often results in a better solution in terms of total throughput, it is unrealistic. In reality, the transport protocol (e.g., TCP) determines the rate of each flow, based on the chosen path and other coexisting flows. Since Internet traffic is hugely dominated by TCP, the model assumes all flows follow the TCP fairness rules. As such, we explore the impact of taking this behavior into account by adding a constraint that approximates the fairness behavior of TCP.

Let O_t be the practical capacity correction factor, which represents the achievable throughput on a technology as a fraction of its theoretical maximum throughput. Its value depends on the transmission technology used by the link, as well as protocol overhead and can be experimentally determined. The TCP fairness constraint is subsequently defined as follows:

$$\forall l \in L, \forall f \in F : \tau_f \cdot \lambda_{l,f} \cdot \sum_{h \in F} \lambda_{l,h} \leqslant O_t \cdot c_{t_l}$$

This constraint, however, contains three decision variables and is thus not linear. Nevertheless, it can be transformed into another set of constraints that can be solved by standard linear programming algorithms [158]. First, a new decision variable needs to be defined:

- $\forall l \in L, \forall f, h \in F : \eta_{l,f,h} = \lambda_{l,f} \cdot \lambda_{l,h}$

The original fairness constraint can then be transformed as follows:

- $\forall l \in L, \forall f, h \in F : \eta_{l,f,h} \leqslant \lambda_{l,f}$
- $\forall l \in L, \forall f, h \in F : \eta_{l,f,h} \leqslant \lambda_{l,h}$
- $\forall l \in L, \forall f, h \in F : \eta_{l,f,h} \geqslant \lambda_{l,f} + \lambda_{l,h} - 1$
- $\forall l \in L, \forall f \in F : \tau_f \cdot \sum_{h \in F} \zeta_{l,f,h} \leqslant O_t \cdot c_{t_l}$

4.2.4 Complexity analysis

In order to have a responsive algorithm that can cope with the highly dynamic network environments, it is important to be able to rapidly solve the MILP. Typically, a branch-and-bound algorithm is used by a solver for solving an MILP. However, upfront different optimizations are performed to reduce the problem size (e.g. presolve or cutting planes). Since the number of constraints in a model has a strong impact on the complexity and solve time, we determine the number of constraints of the formulated MILP.

The flow-based formulation consists of 10 constraints: first, the capacity constraint depends on the number of LCGs (denoted by $|G^L|$). Second, there are 8 flow conservation rules of which 4 depend on the number of flows (denoted by $|F|$), while 3 are depending on the number of flows and the numbers stations. Furthermore, there is a last conservation rule per combination of a flow and a unique pair of two stations. Finally, there is a TCP fairness constraint per combination of each flow and collision group. Assuming the presence of 1 flow per station, the number of stations equals the number of flows, which leads to the following amount of constraints:

$$|G^L| + 4|F| + (3|F| \cdot 3|F|) + |F|^3 + (|G^L| \cdot |F|)$$

4.3 Group-Based Scheduling MILP Formulation

In this section we present a second MILP formulation for the optimization of the network throughput. While this algorithm targets the same heterogeneous networks as before, we now focus on better taking into account the impact of the shared spectrum of wireless technologies and limiting the number of constraints. Furthermore, the previous algorithm (FBSF) takes into account individual links and flows, which can potentially lead to increased execution times for larger network topologies. In contrast, we focus here on the abstraction level of collision groups. This particular formulation is denoted as Group-Based Scheduling Formulation (GBSF) throughout the remainder of this chapter.

4.3.1 Network Model

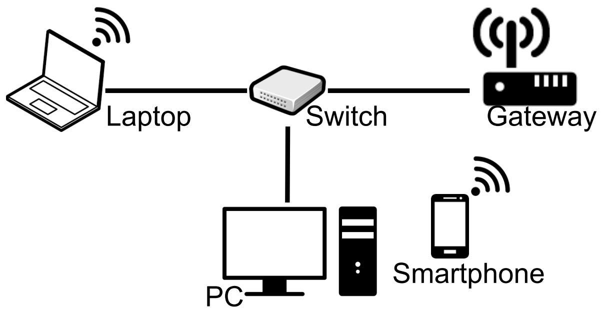
Here, a heterogeneous network is modeled as a multi-graph defined as a tuple (S,G^S) where:

- S is the set of stations $\{s_1, s_2, ..., s_n\}$. These stations represent the different devices within the network, both consumer (e.g., a smartphone or laptop) and infrastructure devices (e.g. a switch or gateway).
- G^S is the set of all Station Collision Groups (SCGs) $\{g_1, g_2, ..., g_n\}$. An SCG $g \in G^S$ is defined as a set of stations $\{s_1, s_2, ..., s_n\}$ with $s_i \in S$ that share a common path to the gateway over a certain technology with a specific capacity. In other words, an SCG encapsulates all the stations that can interfere with each other since they share the capacity of a technology. For a wired technology, like Ethernet, there will be an SCG for each device that is directly connected to the gateway. For instance, all stations that are connected to a switch, with that switch having a single connection to the gateway, share one SCG. For wireless technologies, like Wi-Fi, there is a single SCG per technology per AP, due to the shared medium inherent to wireless technologies. As a consequence of this formulation, where no individual links are taken into account, we only consider single-hop networks. This is in contrast to FBSF that has a more general link-based formulation that also allows for multi-hop networks. Note that this definition of a collision group is different from the one used in Section 4.2.1 as it is here an encapsulation of stations, instead of links.

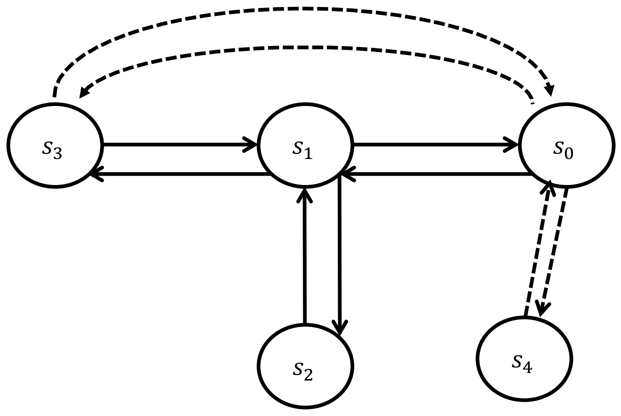
Furthermore, we define the following sets and elements to complete the network model:

- $\forall g \in G^S : \exists c_g \in \mathbb{R}_0$; represents the (theoretical) capacity of the SCG $g \in G^S$.
- $\forall s \in S : G_s : \{\forall g \in G^S \mid s \in g\}$; defines for each station the set of all SCGs to which it belongs (i.e., one for each technology it supports).
- Finally, we also define two subsets to distinguish between different characteristics of technologies:
  - $G_{fdup} \subseteq G^S$ is the set of full-duplex SCGs.
  - $G_{hdup} \subseteq G^S$ is the set of half-duplex SCGs.
  - with $G_{fdup} \cap G_{hdup} = \emptyset$.

  This distinction between full and half-duplex SCGs (and the corresponding underlying technologies) is required because this has a significant impact on the capacity and behavior of these technologies. Ethernet is probably the best known full-duplex technology, while Wi-Fi is usually deployed as half-duplex, with up- and downlink sharing the medium.



(a) Network topology



(b) Multi-graph representation

Figure 4.2: Example of a simple heterogeneous network topology together with its new multi-graph network model representation.

Figure 4.2a shows the same simple heterogeneous network topology as in the previous section. The present devices and the wired and wireless connections are identical to those in Figure 4.1a. However, the network representation shown in Figure 4.2b does differ and contains less elements. Each of the devices is represented by a station $\{s_0, s_1, s_2, s_3, s_4\}$. In this example, there are two SCGs : the first one contains all the stations connected by Ethernet since they share a common link between the gateway and the switch. This first SCG consists of the following stations $\{s_0, s_1, s_2, s_3\}$ and is denoted by the full arrows in the figure. This group is a full-duplex SCG. The second SCG, denoted by the dashed arrows, represents the Wi-Fi connection between the stations $\{s_0, s_3, s_4\}$ and is thus half-duplex. In contrast to the previous model described in Section 4.2.1, we do not model explicitly the separate links and technologies. This simplification helps in minimizing the overall model size and limiting the number of constraints, allowing for applicability in significantly larger scenarios, as we will demonstrate later on.

In addition to the network topology, traffic flows going through the network also need to be modeled. Let us define F as the set of all flows. A flow $f \in F$ is a triple $< s_f, r_f^{in}, r_f^{out} >$ with $s_f \in S$ the station within the network that is the source or destination of the flow within the network, r_f^{in} the incoming desired rate of f $\in \mathbb{R}^+$ and r_f^{out} the outgoing desired rate of f $\in \mathbb{R}^+$. Note that we do assume that the gateway is always one of the two endpoints of the flow, while the other is denoted by s_f. Furthermore, we separate the desired rate of the flow between the incoming and outgoing rates. This allows us to more precisely schedule all flows across the different paths, and it is thus possible that incoming and outgoing packets of a flow are assigned a different route. To clarify, for a TCP flow originating from some web server, the incoming rate is the rate of the data traffic, while the outgoing rate is the one of the ACKs. In our example in Figure 4.2, a flow from the gateway to the laptop, with, for instance, an incoming rate of 10 Mbps and an outgoing rate of 1 Mbps, is represented as the triple $< s_3, 10, 1 >$. Note that in the FBSF algorithm, incoming and outgoing flows are considered as completely separated flows. As such, we now reduce the size of the set F.

4.3.2 MILP formulation

The considered flow scheduling problem is again modeled as an MILP. However, in contrast to the previous formulation in Section 4.2.2, the goal is to maximize the throughput across all the SCGs, instead of calculating and maximizing the rate assigned to individual flows. Note that, consequently, we do not explicitly model the TCP fairness behavior in this formulation. In contrast, among others, we model the specific degradation of the total capacity of a wireless technology, upon increasing numbers of connected devices.

The inputs of the presented MILP consist of the previously described network and flow model. We have thus the following sets as input: S, G^s, G_s, G_{fdup}, G_{hdup}, and F.

Furthermore, we define the following decision variables:

- $\tau_g^{in} \in \mathbb{R}^+$; this variable defines the total incoming rate assigned to an SCG in $g \in G^s$. In other words, a summation over the rates of all incoming traffic flows that will be using the underlying technology of the SCG.

- $\tau_g^{out} \in \mathbb{R}^+$; this variable defines the total outgoing rate assigned to an SCG in $g \in G^s$. In other words, this is identical to the previously defined decision variable, but for outgoing traffic.

- $\lambda_{f,g}^{in} \in \{0, 1\}$; this variable represents the path for the incoming traffic of a flow. If the incoming traffic of flow $f \in F$ is scheduled over an SCG $g \in G^s$, then $\lambda_{f,g}^{in}$ = 1, otherwise it equals 0.

- $\lambda_{f,g}^{out} \in \{0, 1\}$; this variable represents the path for the outgoing traffic of a flow. If the outgoing traffic of flow $f \in F$ is scheduled over an SCG $g \in G^s$, then $\lambda_{f,g}^{out}$ = 1, otherwise it equals 0.

As an objective function, the model maximizes the total rate (bandwidth) of the traffic, both incoming and outgoing, across the entire network:

- $max\left(\sum_{g\in G^s} \tau_g^{in} + \tau_g^{out}\right)$

Finally, we define the following constraints:

- We first define three constraints that make sure the capacity of the individual SCGs and their underlying technologies is not exceeded:
  - For all full-duplex SCGs, the total incoming rate across all flows assigned to the SCG should not exceed the capacity of the group: $\forall g \in G^{fdup} : \tau_g^{in} \leqslant c_g$
  - For all full-duplex SCGs, the total outgoing rate across all flows assigned to the SCG should not exceed the capacity of the group: $\forall g \in G^{fdup} : \tau_g^{out} \leqslant c_g$
  - For all half-duplex SCGs, the total of the incoming and outgoing rate across all flows assigned to the SCG should not exceed the SCG's capacity: $\forall g \in G^{hdup} : \tau_g^{in} + \tau_g^{out} \leqslant c_g$
- Next, we define two constraints that make sure that the rates assigned to an SCG do not exceed the total rate of the traffic that is assigned to the SCG:
  - The total incoming rate for an SCG should not exceed the total amount that is desired by the flows: $\forall g \in G^s : \tau_g^{in} \leqslant \sum_{f\in F} \lambda_{f,g}^{in} \cdot r_f^{in}$
  - The total outgoing rate for an SCG should not exceed the total amount that is desired by the flows: $\forall g \in G^s : \tau_g^{out} \leqslant \sum_{f\in F} \lambda_{f,g}^{out} \cdot r_f^{out}$
- The third group of constraints guarantees the conservation of flows in the network:
  - The incoming traffic for every flow needs to be assigned to exactly one SCG that contains its endpoint station: $\forall f \in F : \sum_{g\in G_{s_f}} \lambda_{f,g}^{in} = 1$
  - The outgoing traffic, for every flow, needs to be assigned to exactly one SCG that contains its endpoint station: $\forall f \in F : \sum_{g\in G_{s_f}} \lambda_{f,g}^{out} = 1$
  - The incoming traffic of each flow, should be assigned to exactly one SCG: $\forall f \in F : \sum_{g\in G} \lambda_{f,g}^{in} = 1$
  - The outgoing traffic of each flow, should be assigned to exactly one SCG: $\forall f \in F : \sum_{g\in G} \lambda_{f,g}^{out} = 1$

4.3.3 Technology capacity estimation

In the previous section, we defined the group-based MILP formulation and introduced the notion of SCGs. These are the encapsulations of all the stations that interfere with each other due to the sharing of the capacity of a certain technology. An important aspect that was not explicitly discussed is the determination of the capacity of the SCGs and their underlying technologies. Since the goal of the algorithm is to optimize the utilization of the network and schedule the flows across the different technologies, the capacity of these technologies should be determined as accurately as possible. The importance of this was already highlighted in Section 4.2.3 where we introduced the practical correction factor O_t to correct the theoretical capacity and acquire a more realistic value. Here, we go one step further by proposing a formulation that takes into account the number of connected devices for that specific technology (and the corresponding infrastructure device). As such we account for the harder determination of the exact capacity of wireless technologies, as this depends, among others, on the number of users, their traffic and location.

To this extent, we define a linear function that will approximate the actual capacity of the different technologies, taking into account the number of stations that are using that particular technology at a certain period in time. For a certain SCG $g \in G^s$, we define:

$$\gamma(g, \alpha, \beta) = \alpha \cdot (\sum_{f \in F} \lambda_{f,g}^{in} + \lambda_{f,g}^{out}) + \beta$$

The parameters α and β are technology specific and we will discuss their determination in Section 4.3.4.

In the three capacity constraints defined in the previous section, we can thus replace the theoretical capacity of the SCG by the function γ:

- $\forall g \in G^{fdup} : \tau_g^{in} \leqslant \gamma(g, \alpha, \beta)$
- $\forall g \in G^{fdup} : \tau_g^{out} \leqslant \gamma(g, \alpha, \beta)$
- $\forall g \in G^{hdup} : \tau_g^{in} + \tau_g^{out} \leqslant \gamma(g, \alpha, \beta)$

4.3.4 Dynamic determination of Alpha and Beta parameters

The actual capacities of the different technologies, in particular for wireless ones, are dependent on several parameters (e.g., the configuration of APs, interference of other devices within or outside the network, and the amount of traffic in the network). Estimating each of these parameters is very challenging and is in some cases a separate research problem (e.g., interference modeling). In order to take these parameters into account, without the need for complex models, we propose an experimental method that can be applied in real-time. For each technology, a series of experiments is conducted where the number of stations and the flow rates are varied, each taking values from a predefined set. For instance, the number of stations can be varied from one up to fifteen, while the traffic rate per station can be varied between the theoretical capacity of that technology and a relatively small value like 1 Mbps to cover both saturated and unsaturated cases. For each number of stations, the achieved rate for all relevant scenarios is averaged and stored. Scenarios where the desired rate of all stations is achieved, are not taken into account since we want to determine the actual maximum capacity of the connection. Afterwards, the list of stored maximum capacities is interpolated (using linear regression) as a function of the number of stations, leading to the function γ, as described above. Note that for Ethernet, due to its full-duplex capabilities, it is sufficient to vary only the traffic as the number of connected nodes does not affect practical link capacity. Therefore, the parameter $\alpha = 0$ for Ethernet. This method can be applied for each heterogeneous environment to capture the specific characteristics and can be rapidly re-executed if needed. Note that similar ideas have recently been adopted by commercial Wi-Fi management solutions, such as Technicolor's Wi-Fi doctor [159].

4.3.5 Complexity analysis

Similar to the previous MILP formulation, we again perform an analysis of the complexity of the proposed GBSF. The group-based formulation contains only 9 constraints of which the first 5 only depend on the number of SCG (denoted by $|G^s|$), and the last 4 only on the number of flows (denoted by $|F|$), if we respect the order of definition in Section 4.3.2. As such, we have the following number of constraints:

$$4|G^s| + 5|F|$$

We can conclude that the presented model has a strongly reduced number of constraints, and thus a lower complexity, in comparison to the previously proposed FBSF. This difference in complexity and the execution time is further compared in Section 4.5.

4.4 Deployment and parameter determination

Our approach selects the optimal route for each traffic flow and thus provides the intelligence that is not present in the current network management solutions themselves. The proposed algorithms (both FBSF and GBSF) are fully independent of the underlying management framework and can be deployed on ORCHESTRA and all of the dynamic flow redirecting solutions listed previously in Section 2.2. However, each solution has its own characteristics and features, some of which can heavily influence the performance or impact of our proposed approach. For instance, our approach can relatively easily be deployed on top of SDN controllers without any changes to client-side devices. However, the available functionality depends on the SDN framework. For instance, with an OpenFlow controller and OVS it is possible to reroute downstream traffic, but not upstream traffic (unless an OVS is installed on the client-side as well). In contrast, the previously discussed

ORCHESTRA framework would be the most appropriate underlying framework as it is the most complete and widely applicable solution. Due to its combination of a centralized controller and VMAC, it is possible to perform coordinated and seamless flow rerouting based on real-time monitoring information. As an alternative, the IEEE 1905.1 standard can be used as it also has a virtual/hybrid MAC that enables the required seamless management features. Both solutions allow for control over both up- and downstream traffic and for distributed intelligence. In the remainder of this section, we discuss how the proposed algorithms can interact with the underlying network management solutions and how the received monitoring information is used to determine or estimate the needed inputs.

4.4.1 Workflow description

When deploying the envisioned approach, a number of different components will interact with each other. For instance, in order to provide the necessary inputs to the algorithms or to roll-out the optimal configuration. Furthermore, it also needs to be determined when precisely an algorithm should be executed. All of these interactions are shown in Figure 4.3. There are three main components (indicated by the different colors): the monitoring, the scheduling of the algorithm, and the execution of the MILP formulation(s). Upon initialization, the monitor loop is started by sending for the first time a request message to all the devices in the network (step 1). Afterwards, in step 2, a time-out is started, indicating the time period in which the stations should respond. In parallel of step 1, upon initialization, the execution of the MILP is scheduled at the latest allowed time (denoted by *max interval*) in step 8. This value denoted the maximum interval between two consecutive runs of the MILP.

When the timeout for the receiving of monitoring information has passed (or when all nodes have responded), the received information is processed and compared with the previously received information (if existing) in step 3. We check for three dynamic events in particular: a changed link state (i.e. a link or technology that has gone down or became available (again)), if new flows have arrived or existing ones have stopped, and if the rate of the flows has changed substantially (denoted by a threshold value). These checks are, respectively, denoted as steps 4 to 6 in Figure 4.3. If at least one such event is detected, the MILP formulation is scheduled to be executed (if possible) in step 12, in order to allow the framework to react to dynamic changes in the network. After checking for such changes, the monitoring loop waits a certain period of time in step 7, denoted by an interval value, before resending requests for monitoring information and restarting the whole process by going back to step 1. Note that this process can be optimized, depending on the available logic and network circumstances, by sending an initial configuration message defining the interval for sending monitoring information. As such, a request for monitoring information does not need to be sent for each interval but only when this parameter should be adjusted. This means that monitoring information is provided in a push-based manner.

The rescheduling of the execution of the MILP (step 12) is only allowed when

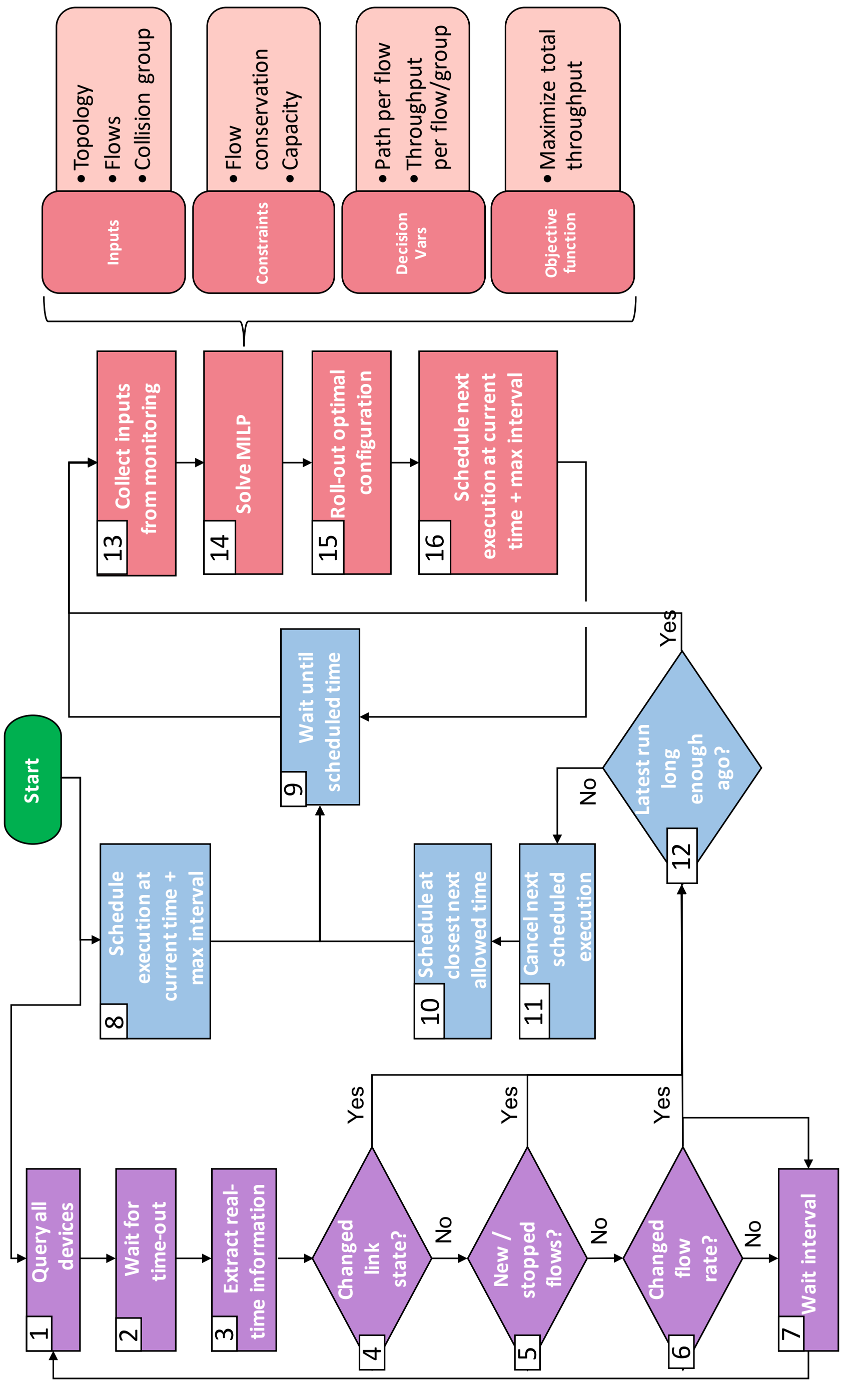

Start
1 Query all devices
2 Wait for time-out
3 Extract real-time information
4 Changed link state?
5 New / stopped flows?
6 Changed flow rate?
7 Wait interval
Yes
No
8 Schedule execution at current time + max interval
9 Wait until scheduled time
10 Schedule at closest next allowed time
11 Cancel next scheduled execution
12 Latest run long enough ago?
13 Collect inputs from monitoring
14 Solve MILP
15 Roll-out optimal configuration
16 Schedule next execution at current time + max interval
Inputs
• Topology
• Flows
• Collision group
Constraints
• Flow conservation
• Capacity
Decision Vars
• Path per flow
• Throughput per flow/group
Objective function
• Maximize total throughput

Figure 4.3: Flow chart illustrating the different steps of the proposed framework.

enough time has passed since the last execution of the MILP, denoted by *min interval*, to avoid unnecessary oscillations and executions. If this interval has already passed, the MILP can be solved immediately (by moving on to step 13). Otherwise, the next execution is rescheduled to the earliest allowed time (steps 10 and 11). Furthermore, upon the execution of the MILP, the necessary information is gathered from the stored monitoring data (step 13), before the allowance of a solver to calculate an optimal configuration in step 14. After calculation, the configuration is rolled-out across the entire network and the next execution is scheduled at the latest allowed time, in respective steps 15 and 16. Note that only devices where the configuration needs to be updated will receive such a configuration message. At the right hand side of Figure 4.3 the different elements of the MILP formulation are depicted.

4.4.2 Estimating flow and network parameters

One of the novelties of our work is the use of real-time monitoring information to estimate the desired flow rates and dynamic network conditions, rather than assuming this information is fully known by the framework. With real-time we mean that the information is processed as soon as it is captured, making it immediately available as feedback. To enable this, as seen in Figure 4.3, our approach relies on a monitoring component that collects the needed information. This monitoring depends on the underlying management framework in place. For instance, ORCHESTRA and SDN solutions natively offer this feature through the reporting of statistics from the managed devices to the controller (e.g., using the OpenFlow protocol). Another option that can be utilized is Link Layer Discovery Protocol (LLDP), a data link layer protocol that allows devices in a network to ask information regarding link quality to their neighbors. Link metric reporting and querying options can thus be defined based on the LLDP, in a similar fashion as is done by the IEEE 1905.1 standard [45]. The typical queryable metrics are the number of packet errors, the amount of transmitted and received packets, MAC throughput, link availability, and theoretical physical rate. Metrics are periodically requested and are valid for a certain amount of time. Note that the designed models currently only require information on MAC throughput, theoretical physical rate, and link state. To summarize, all flow and link parameters (e.g., source, destination, and rate) are estimated in real-time and not known upfront. The network topology is assumed to be known, as the discovery of devices and links is also provided by the link metric reporting mechanism

Because of the fact that we use an estimation (the measured MAC layer rate during the last interval) of the desired rate of a flow, it is possible that the MILP provides a non-optimal solution as it does not know the actual desired rate of the flow (which is application layer information that cannot be known at the network or MAC layers). For instance, if a flow actually desires 12 Mbps while going over a link with a theoretical capacity of 10 Mbps, the framework will only know the measured throughput, which will be lower than the actual desired 12 Mbps. In such a situation, the MILP might decide not to change the path of the flow, while this would have been the case if the actual desired rate was known. This aspect is considered in Section 4.5, where we compare, for a number of scenarios, the performance of the algorithms, both when either estimating the flow rate or knowing the desired flow rate of all present flows.

4.5 Evaluation and discussion

This section evaluates the proposed algorithms using simulation results obtained from the NS-3 event-based network simulator [160]. In the end, we also provide a demonstration of the flow scheduling approach in a small-scale real-life prototype. First, the evaluation setup and scenarios for the simulations are discussed. Afterwards, the algorithms' performance, in terms of achieved throughput and execution time, is evaluated in a variety of static and dynamic scenarios. A comparison to a baseline, using pre-configured interface selection based on hard-coded priorities is provided.

4.5.1 Evaluation setup

In terms of tools, the Gurobi Optimizer (6.5.1) is used to optimally solve the MILP formulations, while the NS-3 event-based network simulator (version 3.25) is used for the simulations [156, 160]. In this simulator, we implemented the entire ORCHESTRA framework, as described in Section 3.2. This allows for the transparent switching of flows between network interfaces, without breaking the end-to-end connection. We assume three technologies present: Ethernet, 5 GHz Wi-Fi, and 2.4 GHz Wi-Fi. In other words, we focus on environments such as LANs. The Ethernet network is constructed of full-duplex Unshielded Twisted Pair (UTP) cables with a theoretical throughput of 100 Mbps. Furthermore, in every network topology considered, there is at least one Ethernet switch present that is connected to the gateway of the network. For the 5 GHz Wi-Fi network a 40 MHz channel is assumed, in contrast to the 20 MHz channel for the 2.4 GHz Wi-Fi network. This allows for a theoretical data rate of respectively 150 Mbps and 72.2 Mbps. To avoid oscillations in the decision-making, we have experimentally determined the following parameters for the scheduling of the algorithm: between two consecutive executions of the MILP formulation should be at least 2 s and at most 10 s. Moreover, the threshold for indicating a changed flow rate is 25 %, while monitoring is reported once every interval of 1 s and the timeout for receiving monitoring information is 0.25 s. Note that besides the generated traffic flows themselves, also the management traffic is considered in the simulations. In other words, the packets that contain monitoring information and configuration instructions, sent between the devices and the controller or vice versa, are also generated and transmitted. As such, our results consider the overhead of management interactions.

In order to generate representative network topologies and conditions, several types of consumer devices are defined, each with different types of interfaces and flows. The device types and their supported interfaces are depicted in Table 4.1.

Device type	Supported network technologies		
	Ethernet	5 GHz Wi-Fi	2.4 GHz Wi-Fi
Desktop PC	×		
Laptop	×	×	×
HD Television	×	×	×
4K Television	×	×	×
Tablet		×	×
Smartphone		×	×

Table 4.1: Overview of the devices in the scenarios with their supported technologies.

Device type	Rate boundaries per flow type		
	Download	Video stream	Video conference
Desktop PC	10–30 Mbps	8–20 Mbps	4–10 Mbps
Laptop	10–30 Mbps	8–20 Mbps	4–10 Mbps
HD Television	5–25 Mbps	10–20 Mbps	5–10 Mbps
4K Television	5–25 Mbps	15–25 Mbps	7.5–12.5 Mbps
Tablet	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps
Smartphone	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps

Table 4.2: Overview of the devices in the scenarios with their supported flow rates.

The exact number of each of these devices is randomly chosen between lower and upper bounds and varies depending on the scenario. Furthermore, three different flow types are defined, as shown in Table 4.2. We assume that the rate of each flow is once again chosen uniformly at random between an upper and lower bound, based on the involved device. Moreover, within the static scenarios, the flow rates do not change over time, while in the other scenarios the download flows will consume as much bandwidth as possible (reflecting their actual behavior). Assuming a static flow rate for the first part of the evaluations, allows us to estimate the difference of using monitored information for flow rate estimation compared to knowing the desired rates. The size of the downloaded file is uniformly at random chosen between 10 MB and 10 GB. We assign one flow per device and as such do not assume the concurrent usage of both Wi-Fi interfaces, as this is generally not supported by current hardware. The different boundaries for the flow rate per traffic and device type, are depicted in Table 4.2. The flow rates were selected based on representative numbers from literature, of existing applications in these three categories [161]. We decided to focus on cases with only TCP traffic, as current Internet traffic is dominated by TCP, despite a slight decrease in the last few years [88]. For example, Lee et al. reported over 95 % of Internet traffic to be TCP in 2010 [87].

Finally, for every described scenario, results are averaged over different randomly generated flow and topology configurations. To this extent, each experiment was repeated 20 times. We also report the standard error for each experiment over these 20 repetitions. As a baseline for comparison, a static configuration is used where all devices are connected to a technology according to the following priorities: Ethernet, before 5 GHz Wi-Fi, before 2.4 GHz Wi-Fi. Furthermore, we compare both the FBSF and GBSF algorithms that have been proposed in, respectively, Sections 4.2 and 4.3. All simulation were done using a single core of an Intel® Xeon® E5-2680 Processor running at 2.8 GHz and with 8 GB RAM.

4.5.2 Parameter values

In order to use realistic values for the bandwidth capacities for the different technologies (both wired and wireless) in our algorithm, a practical correction factor was proposed in Section 4.2.3 and a linear approximation function was defined in Section 4.3.3. First, for the practical correction factor, we experimentally determined the following factors: 0.985, 0.810 and 0.733 for, respectively, Ethernet, 2.4 GHz Wi-Fi, and 5 GHz Wi-Fi. To determine these factors, we ran multiple experiments per technology over a link between two nodes, using the network settings outlined above. These experiments compared the actual throughput with the theoretical maximum under different circumstances: varying number of flows (between one and four), and varying flow rates (between 8 Mbps and 120 Mbps). Second, using a similar network setup we also determined the parameters α and β per technology for the approximation functions. For Ethernet, 15 experiments were conducted over a link between two nodes with s varying number of flows and their rates: starting from one flow up to 15 and varying their rates between 8 Mbps and 120 Mbps. As such, both unsaturated and saturated scenarios were evaluated. We report the following values: α is zero (due to the full-duplex character) and β is 0.9854 multiplied by the theoretical capacity of the Ethernet link (in this case 100 Mbps). For the two different Wi-Fi technologies, the experiment was repeated 30 times per technology. Besides flow count and rates, we also evaluated the impact of an increasing number of stations starting from one up to fifteen. The obtained values are, for α and β respectively: for 2.4 GHz Wi-Fi -1.7413 and 57.578, and for 5 GHz Wi-Fi -3.2085 and 112.99.

4.5.3 Home and office scenarios

To demonstrate the impact of our framework in LANs, two basic scenarios were created that reflect two typical representative environments: a home and an office network. For each setup, the exact number of devices per type and the possible flows are depicted in Table 4.3. The results for the home and office scenario are shown in Figure 4.4. The graphs compare the static baseline and the two proposed load balancing algorithms FBSF and GBSF to the total sum of the desired flow rates. Both scenarios show a strong improvement in the total throughput when using the GBSF algorithm as compared to the baseline and FBSF.

For the home scenario, an average throughput of 117.01 Mbps ($\pm$0.66) for the baseline is achieved. In contrast, the average throughput with GBSF increases to 137.62 Mbps ($\pm$3.16). This means that there is an increase of 20.62 Mbps or

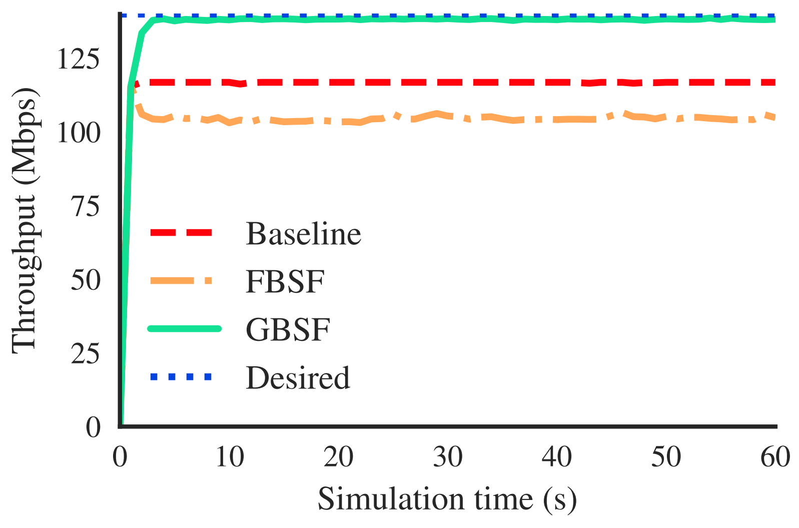
Device	**Home**	**Office**	**Flows**
Switches	1	3	N/A
Desktop PC	1	11	Download
Laptop	3	7	Download / Video conference
4k TV	2	1	Video stream
Tablet	1	0	All types of flows
Smartphone	3	4	All types of flows
Total devices	10	23	

Table 4.3: Topology parameters for the two scenarios.

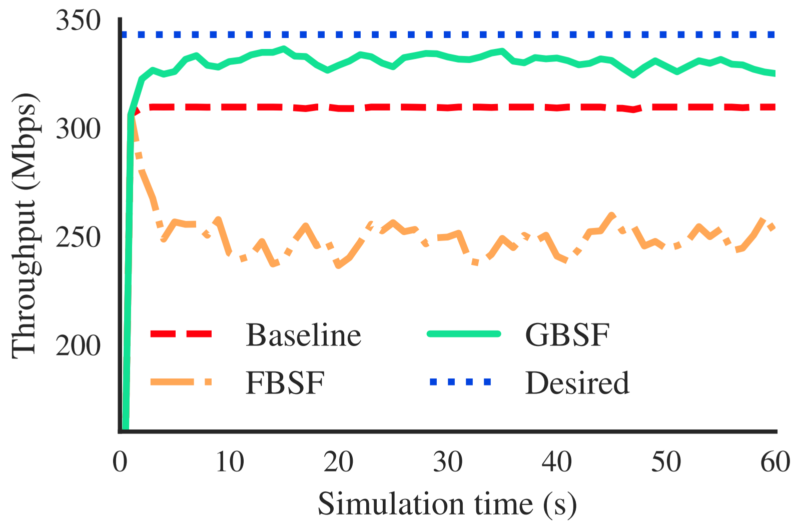
17.62 %. Figure 4.4a shows that GBSF approximately reaches the desired traffic rate for all flows combined. The difference between the framework and the total desired rate is only 1.72 Mbps or 1.23 %. Figure 4.4a also indicates that FBSF does not perform well and even performs worse than the baseline. The reason for this lays in the fact that FBSF does not take into account the characteristics of the wireless spectrum as detailed as GBSF and ignores the impact of interference between different stations. Note that while performing this evaluation, we noticed that the performance of FBSF is higher, and even offers an improvement towards the baseline, for smaller scenarios. In these smaller scenarios, the capacity of the Ethernet links is limited to 10 Mbps and the rates of the flows were more than half the size of the current values. However, due to the fact that these values are rather unrealistic, these results are omitted.

For the office scenario, an average throughput of 310.39 Mbps ($\pm$2.67) is recorded for the baseline, while the average throughput with GBSF increases to 329.52 Mbps ($\pm$3.03). This means that there is an increase of 19.13 Mbps or 6.16 %. Despite this improvement, one can see in Figure 4.4b that GBSF does not reach the desired rates for all flows. The difference between the framework and the total desired rate is 12.40 Mbps or 3.63 %. However, the reasons for this can be found outside of our framework. First, this scenario is slightly oversaturated. As such, the physical capacities of the network are being reached. Second, since an approximation of the capacities of the wireless technologies is used, a slight inaccuracy is possible. Third, because of the oversaturation, it is also harder to estimate the actual desired flow rate, as discussed in Section 4.4.2. At the end of this section, we discuss this in more detail. Figure 4.4b also clearly shows that FBSF fails to provide any improvement in comparison to both baseline and GBSF.

For both scenarios, the increase in throughput offered by GBSF is made possible by scheduling the different traffic flows more appropriately across all technologies. Table 4.4 shows, for both the home and office scenario, the average load distribution of flows across the three technologies for the different algorithms. For the home scenario with 20 flows we see how for the baseline the Ethernet technology is the bottleneck. This is explained by the static configuration of the baseline where Ethernet is prioritized over the 5 GHz Wi-Fi network, while the 5 GHz Wi-Fi network is prioritized over the 2.4 GHz Wi-Fi network, as stated in



(a) Home scenario



(b) Office scenario

Figure 4.4: Throughput as a function of time for different scenarios, comparing our proposed algorithms to the static interface selection baseline.

	Algorithm	Technology load distribution		
		Ethernet	5 Ghz Wi-Fi	2.4 Ghz Wi-Fi
Home Scenario	Baseline	16 flows	4 flows	0 flows
	GBSF	7 flows	7 flows	6 flows
	FBSF	2 flows	11 flows	7 flows
Office Scenario	Baseline	56 flows	8 flows	0 flows
	GBSF	44 flows	12 flows	8 flows
	FBSF	37 flows	16 flows	11 flows

Table 4.4: Overview of the distribution of flows over the different technologies for the home and office scenarios.

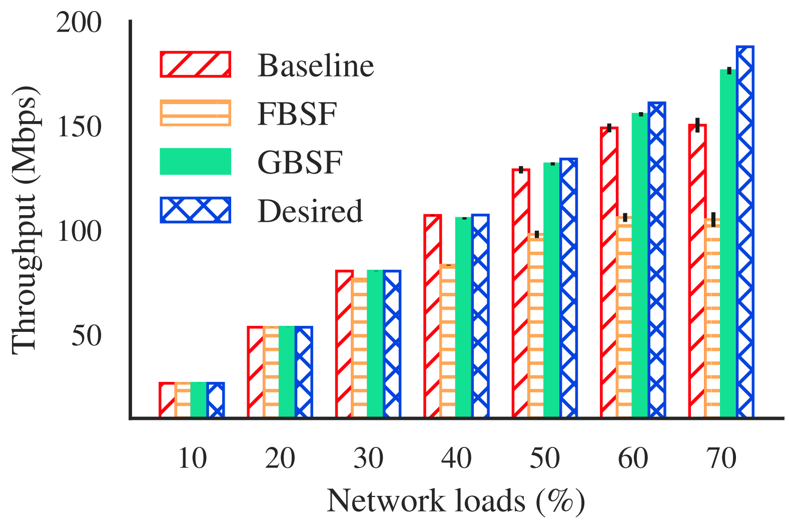
Section 4.5.1. In contrast, GBSF succeeds in scheduling the flows optimally to increase the network-wide throughput: on average, 7 flows are scheduled on Ethernet, 7 flows on 5 Ghz Wi-Fi, and 6 flows on the 2.4 Ghz Wi-Fi. On the other hand, FBSF overloads the two Wi-Fi networks, as only 2 flows are scheduled on Ethernet. For the larger office scenario with in total 64 flows, similar load distributions are obtained.

Furthermore, the impact of the implementation of the VMAC layer can clearly be noticed as the actual technology switching of flows cannot be noticed in the figure. The execution times for GBSF for the home and office scenario are, respectively, 8.16×10^{-4} s ($\pm 2.62 \times 10^{-5}$) and 1.84×10^{-3} s ($\pm 1.96 \times 10^{-3}$). For FBSF the execution times are higher, respectively, 0.11 s ($\pm 3.99 \times 10^{-3}$) and 5.61 s ($\pm$1.79). This clear difference in execution time shows an improvement in performance of GBSF, compared to FBSF. This is discussed in more detail in the next section.

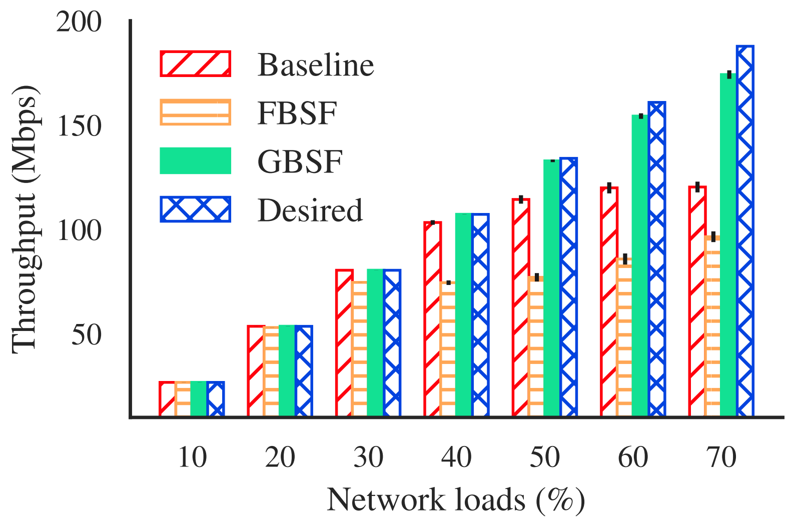
In order to further assess the impact of the unknown desired flow rates, each randomly generated scenario was executed a second time using the actual, rather than the estimated, desired flow rates. For the home scenario nearly no difference can be identified, while a slightly higher throughput can be noticed for the office scenario: for the home scenario the average overall throughput for GBSF equals 137.64 Mbps ($\pm$3.15). This is only 0.02 Mbps higher than the throughput achieved when using the real-time measured rate for each flow. In the case of the office scenario, where the standard GBSF formulation does not reach the desired rate, the impact of using the actual desired rates of the flows is more clear as it results in an average throughput of 331.50 Mbps ($\pm$3.03), which is 1.98 Mbps higher than the standard GBSF. This means that knowing the desired flow rates would offer an additional improvement of 0.58 % in the larger office scenario.

4.5.4 Impact of network load and scalability

In order to further investigate the performance of the MILP formulation, we conducted experiments with varying network loads. This is achieved by generating a set of devices, each with a uniform randomly assigned flow that has, in turn, a



(a) Random types of devices



(b) Only Ethernet-enabled devices

Figure 4.5: Throughput as a function of network load, error bars depict the standard error.

randomly chosen type and rate. The total desired rate of all flows equals a certain percentage of total theoretical network capacity. Experiments were performed for loads of 10, 20, 30, 40, 50, 60 and 70 % of the theoretical network capacity. Furthermore, two scenarios were evaluated. First, using all types of devices, with every device selected at random from the set of seven devices listed in Table 4.1. For the second scenario, only devices that at least have an Ethernet interface were selected. In other words, this scenario does not contain smartphones and tablets. This way more traffic will initially pass over the Ethernet links, making it easier to reach a saturated state on that interface.

Figure 4.5 illustrates that GBSF offers an increase in throughput under the higher network loads. As the network load increases, so does the relative performance of our proposed framework compared to the baseline, for both scenarios. Furthermore, FBSF again fails to reach the level of the baseline.

For the first scenario, depicted in Figure 4.5a, from a network load of 50 % onwards, GBSF start having a noticeable impact. The best result is noticed in the case of a network load of 70 %. Here the baseline achieves an average total throughput of 150.33 Mbps (±3.53), while running GBSF leads to an throughput of 176.42 Mbps (±1.16). In other words, there is an increase of 26.09 Mbps or 17.35 %. The achieved rate of GBSF is 11.43 Mbps or 6.09 % below the desired rate. On the other hand, FBSF only achieves 105.64 Mbps (±2.88) or 70 % of the throughput of the baseline.

For the second scenario with only Ethernet-enabled devices, depicted in Figure 4.5b, GBSF provides an improvement from a network load of 40% onwards. The biggest improvement can again be seen with a network load of 70% . We can notice an improvement from 120.38 Mbps (±2.57) for the baseline to 174.25 Mbps (±2.04) for GBSF. This is an increase of 53.87 Mbps or 44.75 % and is only 13.56 Mbps or 7.22 % below the desired rate. FBSF is consistent in its underperformance as it only achieves a throughput of 91.86 Mbps (±4.64).

Similar to the previously discussed home and office scenarios, we also compared the results of the framework with and without knowing the desired flow rates upfront. A similar impact occurs when evaluating the home and office scenarios. For the first scenario with all types of devices, the usage of the proposed framework (GBSF) with knowledge of the rates results in a throughput of 179.99 Mbps (±1.16) for the scenario with a load of 70% . This is an increase of 3.57 Mbps compared to the normal execution. For the second scenario with only Ethernet-enabled devices, running GBSF with the rates of the flows known, results in a throughput of 177.05 Mbps (±2.17) for the same scenario. This is an increase of 2.80 Mbps with respect to the standard execution.

Next, we evaluated the time it takes to execute the MILPs and find the optimal configuration. Table 4.5 shows the averages of the measured values for the different runs for the first scenario. The table offers a comparison between the execution times of both FBSF and GBSF, to the number of flows present across the different network load percentages. The results, as depicted in Table 4.5, show that the execution times of both algorithms scale exponentially. However, there is a noticeable improvement in terms of execution time of GBSF, in comparison

Network Load (%)	**Flows**	**FBSF execution time (±SE)**	**GBSF execution time(±SE)**
10	4	0.01 s (±0.01)	1.89 $\times 10^{-4}$ s (±1.92 $\times$ 10^{-4})
20	7	0.04 s (±0.01)	5.06 $\times 10^{-4}$ s (±1.11 $\times$ 10^{-3})
30	11	0.11 s (±0.03)	7.74 $\times 10^{-4}$ s (±1.63 $\times$ 10^{-3})
40	13	0.25 s (±0.14)	9.29 $\times 10^{-4}$ s (±6.84 $\times$ 10^{-3})
50	16	0.43 s (±0.26)	1.17 $\times 10^{-3}$ s (±1.01 $\times$ 10^{-2})
60	20	0.81 s (±0.50)	1.33 $\times 10^{-3}$ s (±8.75 $\times$ 10^{-3})
70	23	1.46 s (±0.71)	1.75 $\times 10^{-3}$ s (±1.33 $\times$ 10^{-3})

Table 4.5: Comparison of the execution time of the proposed FBSF and GBSF algorithms.

Network Load (%)	**Flows**	**FBSF execution time (±SE)**	**GBSF execution time(±SE)**
10	7	0.03 s (±0.02)	2.88 $\times 10^{-4}$ s (±1.36 $\times$ 10^{-4})
20	12	0.18 s (±0.06)	6.63 $\times 10^{-4}$ s (±3.90 $\times$ 10^{-4})
30	19	0.80 s (±0.33)	1.06 $\times 10^{-3}$ s (±1.85 $\times$ 10^{-2})
40	24	1.80 s (±0.64)	1.28 $\times 10^{-3}$ s (±5.33 $\times$ 10^{-2})
50	31	4.10 s (±2.16)	1.58 $\times 10^{-3}$ s (±3.44 $\times$ 10^{-2})
60	38	8.68 s (±2.99)	3.07 $\times 10^{-3}$ s (±2.15 $\times$ 10^{-1})
70	44	17.49 s (±8.26)	2.88 $\times 10^{-2}$ s (±7.79 $\times$ 10^{-1})

Table 4.6: Comparison of the execution time of the proposed FBSF and GBSF algorithms, under an increased number of flows.

with FBSF. The difference is the highest for a network load of 70 % where it takes only 1.75 $\times 10^{-3}$ s to optimally solve the MILP formulation. This improvement, as explained in Section 4.3, is due to the change of objective function where the goal is to maximize the throughput across all the SCGs, instead of calculating and maximizing the rate assigned to individual flows. To illustrate this further, we rerun the first network load scenario (with randomized devices) but we halve the lower and upper bounds for the rates of all flows. This way we artificially, approximately, double the number of flows, for each percentage of the network load. The results in Table 4.6 show more clearly the increased execution time of FBSF in terms of the number of flows. For a network load of 70 % it takes on average 17.49 s (±8.26) to execute FBSF. This is in strong contrast with GBSF that only takes 2.88 $\times 10^{-2}$ s (±7.79 $\times$ 10^{-1}).

To further investigate the scalability of the proposed solution, we conducted an experiment where we evaluated execution time in terms of the number of stations and SCGs (i.e., technologies). We perform this evaluation only for GBSF, as this formulation outperformed FBSF significantly in the previous experiments. Since the scalability of the NS-3 simulator is limited, as each packet is actually generated, we conducted a number of emulations on an Intel NUC. We artificially provide the necessary inputs to the framework, thereby varying the number of stations

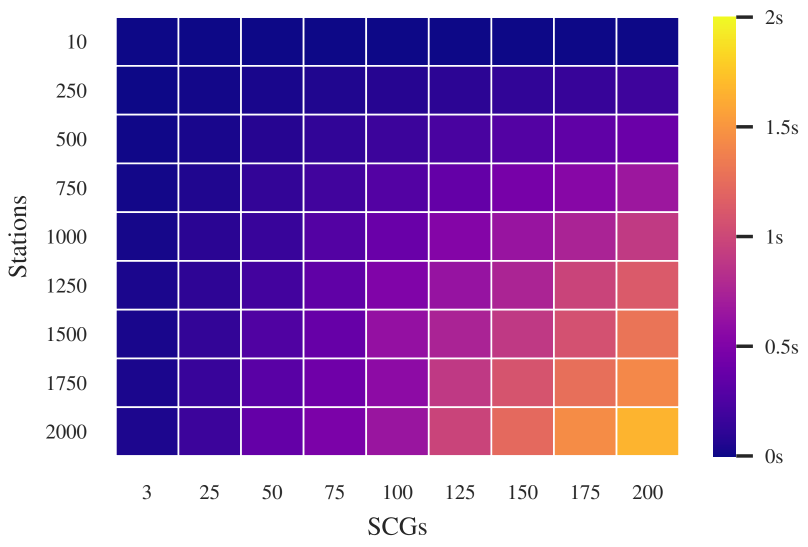


Figure 4.6: Scalability of GBSF in terms of stations and SCGs.

between 10 and 2000 and the number of SCGs between 3 (as in the simulations) and 200. Furthermore, the same assumptions as stated in Section 4.5.1 apply. For each pair of the number of stations and SCGs, we take the average across 20 executions, each with a randomly generated topology. Figure 4.6 shows the resulting heatmap, where every colored cell indicates the average time to solve the MILP for a specific pair. We can see that the execution time does increase when the number of stations and SCGs rises, but stays under the 2 s for all configurations. In particular, it takes on average 1.67 s ($\pm$ 0.01) to solve the MILP for the largest configuration.

We summarize that the results confirm that the proposed GBSF improves in terms of scalability and performance. The formulation can thus be solved fast enough to react to dynamic network changes, even in large and dense networks.

4.5.5 Dynamic scenarios

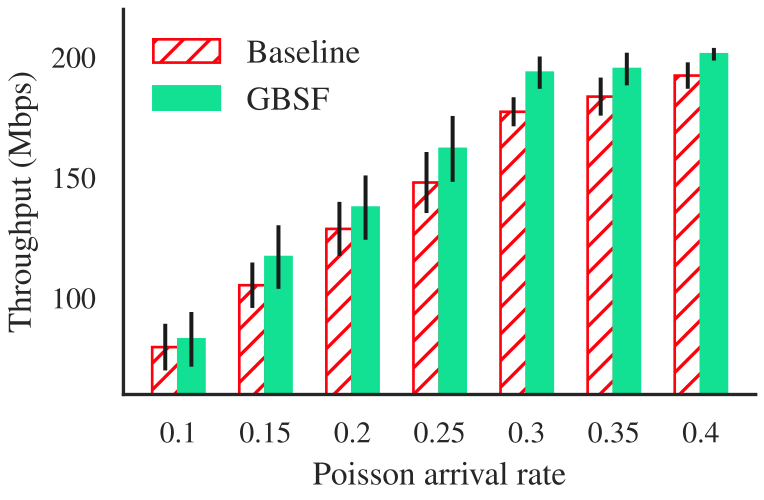
So far we have only considered scenarios with static flow rates and arrival times, as we assumed the flows to be present throughout the entire simulation. While this helped in determining the performance of the framework, this is not always realistic. Furthermore, an important performance metric of the framework is its adaptability to dynamic conditions. To evaluate this aspect, we consider such a dynamic scenario in this section. All download flows act as in reality and consume as much bandwidth as possible (or assigned to them by the TCP rate control algorithm), until the desired amount of data has been downloaded (or the maximum flow length has been reached). Moreover, flows arrive according to a Poisson distribution and the flow length is uniformly at randomly chosen between bounds. The impact of different arrival rates is evaluated (Poisson parameter λ) for two scenarios with different flow lengths. They are randomly chosen between 5 and 15 s for the first scenario. For the second one, the bounds are doubled to 10 and 30 s. Note that we only evaluate the performance of GBSF based on the results of the previous experiments.

Figure 4.7 shows the results for both scenarios. For all different arrival rates, the framework outperforms the baseline significantly. For the scenario with the, on average, smaller flow lengths, depicted in Figure 4.7a the improvement is quite consistent, with a slightly smaller improvement for parameter values of 0.1 and 0.4. It is also clear that, in both scenarios, for the higher arrival rates, both the baseline and framework reach the physical limits of the network. For the second scenario, with longer flow lengths, the highest improvements by the framework can be noticed for the arrival rates of 0.15 en 0.2. For the arrival rate of 0.15, there is an increase from 131.71 Mbps ($\pm$11.09) to 150.10 Mbps ($\pm$13.37), or 13.96 %. For the parameter value of 0.2, there is an increase from 148.26 Mbps ($\pm$8.70) to 173.30 Mbps ($\pm$8.86), or 16.89 %. It makes sense that these parameters result in the highest increase because for the smallest values there is not enough traffic to benefit from the improvements of the framework. On the other hand, for the higher arrival rates with more overall traffic, the network capacity is a limiting factor. Finally, we note that the standard error is quite large, indicating large differences between the different individual experiments.

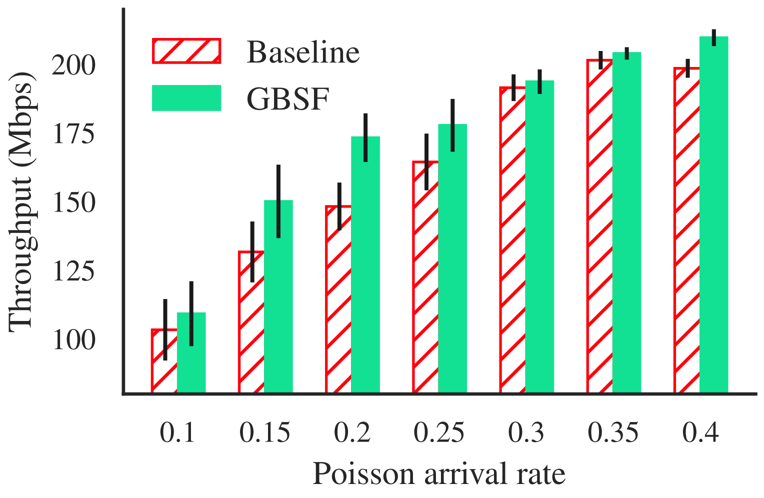
4.5.6 Impact of link failure

In a final scenario, we investigated the impact of a link failure, as this is a very disruptive and plausible network event. For the topology we use the same setup as for the home scenario, defined in Section 4.5.3 but we only replace one device (an HD-TV replaces the PC). In this scenario we assume that the Ethernet connection between the switch and gateway fails between the timestamps of 20 s and 40 s. In order to get a representative baseline, we did a number of experiments to determine realistic behavior upon link failure. We disconnected the Ethernet connection and measured the time it takes to switch to Wi-Fi on a standard MacBook Air (13-inch, early 2015). Afterwards, when the Ethernet connection was up again, we measured how long it took the operating system to switch back to Ethernet. It turned out that the disconnection is recognized almost immediately (below a second) and no real disruption could be noticed. However, upon Ethernet reconnection, it took on average at least 8 s before traffic was switched back from Wi-Fi to Ethernet. Our baseline will thus act correspondingly with an immediate handover to Wi-Fi upon link failure and a delayed handover (of 8 s) in the other direction.

The averaged results of this scenario are shown in Figure 4.8. For the baseline, the impact of the link failure can clearly be noticed because of a drop in the total throughput of more than 50 %. In strong contrast, for the proposed approach, this drop is limited to less than 20 %. This difference is explained by the fact that



(a) Short length of flow



(b) Long length of flow

Figure 4.7: Throughput as a function of Poisson parameters, error bars depict the standard error.

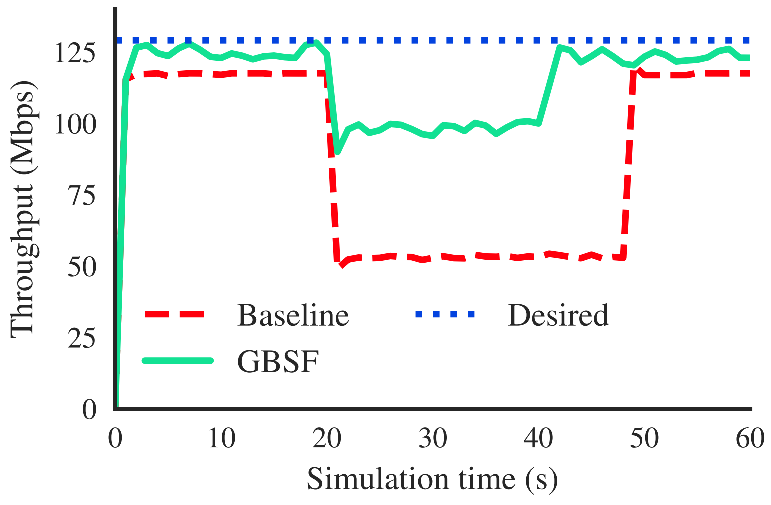


Figure 4.8: Throughput as a function of time for a scenario where a link failure occurs at time 20 s.

our algorithms schedules the traffic more evenly across both links. As such, it is more robust against a link failure. While the desired rate, before the link failure, is almost reached by the algorithm, this is not possible anymore after the handover because the capacity of the two Wi-Fi interfaces is simply not large enough. It is also clear that when the Ethernet interface returns, the framework almost immediately reschedules the flows across all three interfaces. Thus reaching the same level of throughput as before the link failure. For the baseline, as discussed before, it takes 8 s before the traffic is switched back. This behavior results in an average throughput of 82.12 Mbps (±0.60) for the baseline and an average throughput of 113.54 Mbps (±2.36), compared to a desired rate of 129.14 Mbps (±3.38). In other words, deploying the proposed approach introduces an improvement of 38 %.

4.5.7 Prototype

We have thus far evaluated the proposed approach under numerous scenarios in a simulation environment. To further investigate performance, a real-life prototype was developed. The architecture of this prototype, as shown in Figure 4.9, consists of two client devices with two interfaces. Instead of opting again for the ORCHESTRA framework as the underlying network management solution, we show the versatility of the proposed flow scheduling approach by utilizing an SDN setup. First, each device has an Ethernet connection to the gateway over a single switch. In other words, the physical link between gateway and switch is shared. Second, both client devices also have a 2.4 GHz Wi-Fi connection. The rest of the

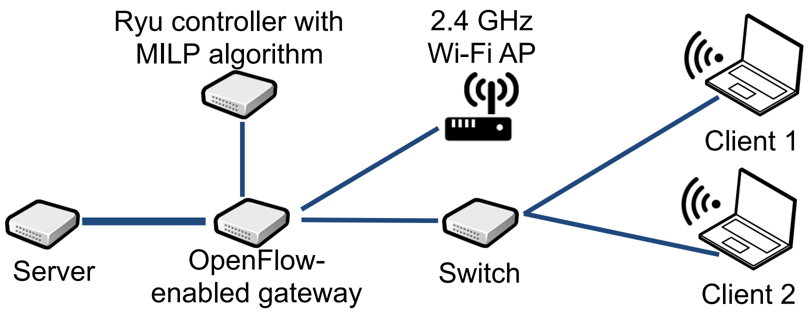
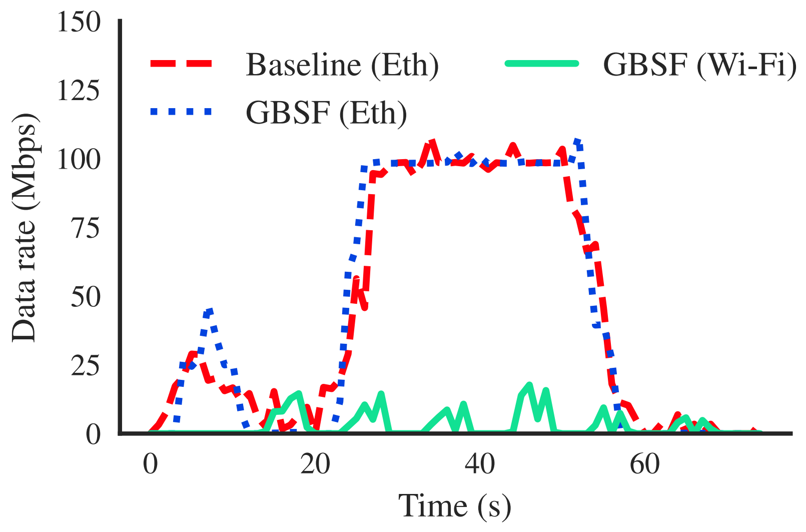


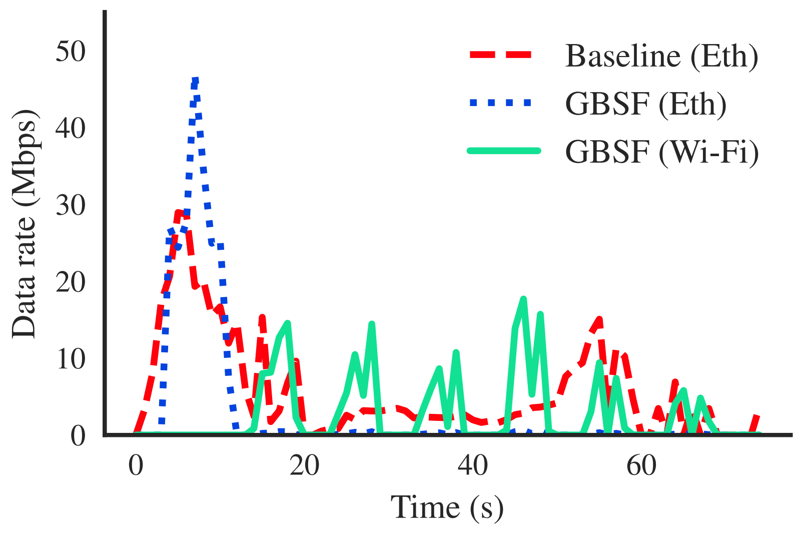
Figure 4.9: Architecture of the prototype setup

architecture consists of a server and an OpenFlow-enabled gateway. On this gateway, we have installed OVS, a virtual multi-layer switch that is controllable by the OpenFlow protocol. The installation of an OVS allows us to perform transparent handovers between the different available technologies, in this case, Ethernet and 2.4 GHz Wi-Fi. For the SDN controller we have opted for the Ryu implementation since it is open source and is being actively developed. On this controller, we deployed our intelligence that decides if and when a flow should be rerouted. For the client devices, regular ASUS laptops are used. For the AP we used an off-the-shelf Netgear N750, while all other devices are Intel NUCs. All Ethernet connections are limited to 100 Mbps, except for the connection between the server and gateway that has a capacity of 1 Gbps (as indicated by the slightly thicker line in Figure 4.9.

We consider the following scenario: the first client is watching a real-time video stream, while after roughly 20 s the second client starts a download that lasts for approximate 40 s. The GBSF algorithm is compared to a static baseline that assumes that all traffic uses the Ethernet connections, as this is prioritized over the wireless ones. For both the baseline and algorithm case, five different runs are conducted and results are averaged. Figure 4.10 indicates that GBSF spreads the traffic across the two technologies, as the video traffic is switched to the Wi-Fi route, while the download flow remains on Ethernet. Figure 4.10a shows the impact for all traffic on both interfaces in terms of data rate, while Figure 4.10b illustrates the impact on the video flow that is switched. The use of the flow scheduling approach results in an average overall data rate of 45.54 Mbps ($\pm$5.37), in respect to the baseline of 40.66 Mbps ($\pm$4.77). In other words, there is an increase of 4.88 Mbps or 12.01 %. This increase can clearly be seen in Figure 4.10b, where the video traffic on the Wi-Fi interface continues its burst behavior after the background traffic is introduced. In contrast, in the baseline scenario, the video traffic is not able to reach its desired rates, until the background traffic is removed (around 55 s). Even more important than the improvement in data rate is the improvement in terms of user experience: in the case of the baseline, the video stream is not able to reach its required bandwidth causing the buffer of the client's video player to drain and thus introducing freezes to the video. This stands in strong contrast to the results



(a) All traffic



(b) Video traffic only

Figure 4.10: Throughput over Ethernet and Wi-Fi interfaces as a function of time for an illustrative scenario on the real-life prototype.

of the framework, where at no point in time any impact on the video or its buffer can be noticed, not even when the handover between the two technologies is executed. Furthermore, we can also report that on average the Gurobi solver needed 3.00×10^{-4} s ($\pm 1.7407 \times 10^{-5}$) to optimally execute the MILP. This is in line with the reported execution times for the simulations.

4.6 Conclusion

In this chapter, we have introduced a dynamic flow management approach that can be deployed on top of multi-technology management solutions such as ORCHESTRA or the IEEE 1905 standard. Based only on real-time monitoring information, the approach aims to optimize the scheduling of traffic flows to increase network-wide throughput. The notion of collision groups is introduced to capture the shared medium of wireless networks where contention among users may affect network capacity. An evaluation of the MILP formulations, through a combination of extensive NS-3 simulations and a real-life prototype evaluation, shows that the presented approach can indeed react in real-time to dynamic network changes and offers a significant improvement in terms of throughput. Across different scenarios, an increase in throughput of around 20 % can be noticed compared to a static baseline. Higher increases of up to 40 % can be achieved in some scenarios, for instance, under the presence of link failures.

5 Scalable load balancing and flow management for mobile heterogeneous wireless networks

"You were the chosen one! It was said that you would destroy the Sith, not join them, bring balance to the force, not leave it in darkness."

–Obi-Wan Kenobi (Star Wars: Episode 3 - Revenge of the Sith, 2005)

The contributions presented in this chapter are based on the publication titled *"Scalable Load Balancing and Flow Management in Dynamic Heterogeneous Wireless Networks"*.

5.1 Introduction

In Section 1.1, we showed how the number of connected devices has known a massive increase in the last decade. The two main reasons for this (r)evolution are the worldwide adoption of mobile and IoT devices. These devices have further increased the heterogeneity in the wireless networks due to their very diverse characteristics in, for instance, moving speed or mobility patterns (e.g., the difference between a connected car and a laptop). This on top of the already existing management puzzle raised by the different demands and characteristics of consumer devices and communication technologies, as discussed in the previous chapter. Furthermore, mobile devices can have a significant impact on network performance. Certain mobility patterns can cause a large number of devices to connect to the network at (nearly) identical positions, overloading certain locations in the network infrastructure. Especially, since all devices typically connect to the closest APs or base station, based on signal strength.

In order for a mobile device to remain connected to the Internet, it will typically need to perform multiple handovers between different infrastructure devices or even technologies, along the length of its route. For instance, recall a scenario where you are watching a live video on your smartphone or tablet and have to leave the house. In order to continue watching the video, a handover between the in-home Wi-Fi network and the cellular networks is likely appropriate. In Chapter 3, we already presented the ORCHESTRA framework that enabled seamless inter- and intra-technology handovers for devices that are equipped with the proposed VMAC. Furthermore, in the Chapter 4, we introduced intelligence that could be deployed on top of network management solutions, such as ORCHESTRA, in order to optimize the APs selection and flow scheduling across the entire network. The proposed algorithms are able to reroute flows across different wired and wireless connections. However, while these algorithms are able to capture the specific wireless context (in particular the shared medium), they did not take the mobility of stations into account. Furthermore, as explained in Section 2.3, existing approaches from the state-of-the-art are technology depended and are based on theoretical models that assume full knowledge about traffic and network parameters.

To this extent, we extend our work presented in Chapter 4 in several ways. First of all, the previous FBSF and GBSF algorithms focused exclusively on flow scheduling and traffic offloading between wired and wireless links (as they are typically found in LANs). As such, the mobility of connected devices was not modeled. In contrast, we now focus specifically on wireless technologies and allow for multiple wireless infrastructure devices. Furthermore, we take the mobility of users and the distance between connected devices and the network infrastructure into account. In particular, we present three different algorithms: first, an optimal MILP formulation is presented, taking into account the aforementioned elements. These new elements, consequently, extend the complexity of the mathematical formulation, which has a dramatic impact on execution time and scalability. Therefore, we present two different heuristics that ensure scalability by sequentially solving the different parts of the problem, instead of solving it globally.

The remainder of this chapter is structured as follows. First, we introduce the mathematical problem formulation in Section 5.2. This section also discusses, among others, the network model and the deployment of the load balancing algorithms. Next, in Section 5.3 we propose two different heuristics to improve scalability. Finally, we present a thorough evaluation in Section 5.4, where we compare the three different algorithms across multiple scenarios.

5.2 Multi-technology load balancing formulation

In this section, we first introduce a model for heterogeneous wireless networks and present an MILP formulation representing the load balancing problem. Afterwards, we discuss in detail the interaction with the network and how certain parameters can be determined.

5.2.1 Network model

A heterogeneous wireless network is modeled as a multi-graph defined as a tuple (S,T,B) where:

- S is the set of stations $\{s_1, s_2, ..., s_n\}$. These stations represent all kinds of connected devices, depending on the modeled network (e.g., smartphones, sensors, vehicles).
- T is the set of technologies $\{t_1, t_2, ..., t_n\}$. This can, for instance, be Wi-Fi (e.g., IEEE 802.11ac, IEEE 802.11ad, ...) or LTE.
- B is the set of all Basic Service Sets (BSSs) $\{b_1, b_2, ..., b_n\}$. A BSS is defined as a set of stations $\{s_1, s_2, ..., s_n\}$ that are connected, over a specific technology, to an AP, an LTE base station, or an equivalent infrastructure device. In other words, a BSS encapsulates all the stations that can compete with each other since they share the capacity of a technology. We assume no interference between BSSs that are in the range of each other (i.e., the use of different non-overlapping channels). Note that the concept of a BSS is identical to that of an SCG in the previous Chapter 4, but the naming is more appropriate within the context of wireless networks.

Furthermore, we define the following sets and elements:

- $\forall s \in S : T_s$; defines per station the set of all technologies $t \in T$ that are supported by that particular station.
- $\forall b \in B : B_t$; is the set of all BSSs that offer a certain technology $t \in T$.
- $\forall s \in S : B_s$; is the set of all BSSs to which a station $s \in S$ can belong. In other words, these are all the BSS of which the infrastructure device (e.g., AP) is in range of the station (for a supported technology).
- Finally, we define $d_{s,b}$ and $b_{s,b}$ to be, respectively, the data rate (depending on the MCS) and bit error rate of the station $s \in S$ for a specific BSS $b \in B$. These values depend on the mobility and position of stations, with respect to each BSS, and can change heavily over time. We discuss the estimation of $d_{s,b}$ and $b_{s,b}$ later on (in Section 5.2.3).

In addition to the network topology, we also need to model traffic flows going through the network. Therefore, we define F as the set of all flows. A flow $f \in F$

is a triple $< s_f, r_f^{in}, r_f^{out} >$ with $s_f \in S$ the station within the network that is the source or destination of the flow within the network, r_f^{in} the incoming desired rate of f $\in \mathbb{R}^+$ and r_f^{out} the outgoing desired rate of f $\in \mathbb{R}^+$. Note that we do assume that the entry to the network (e.g., a gateway) is always one of the two endpoints of the flow, while the other is denoted by s_f. Furthermore, we separate the desired rate of the flow between the incoming and outgoing rates. This allows us to more precisely schedule all flows across the different paths, as incoming and outgoing packets of a flow can be assigned a different route. To clarify, for a TCP flow originating from some web server, the incoming rate is the rate of the data traffic, while the outgoing rate is the one of the ACKs. In the case of a UDP flow originating from the same web server, the outgoing rate will be 0 as there are no ACKs.

5.2.2 MILP formulation

The considered multi-technology load balancing problem is modeled as an MILP, which consists of the necessary inputs, decision variables, an objective function, and a set of constraints. The inputs of the presented MILP consist of the previously described network and flow model. Additionally, we need one more input: we define χ_b to be a linear function that approximates the capacity of the different BSSs, taking into account the number of stations and the particular technology. This corresponds to the linear approximation function that was defined within the context of the GBSF algorithm (cf. Section 4.3.3). This is further discussed in Section 5.2.3.

Next, we define the following decision variables:

- $\tau_f^{in} \in \left[0, r_f^{in}\right]$; the total incoming rate assigned to a flow $f \in F$.
- $\tau_f^{out} \in \left[0, r_f^{out}\right]$; the total outgoing rate assigned to a flow $f \in F$.
- $\lambda_{f,b}^{in} \in \{0,1\}$; the path for the incoming traffic of a flow. If the incoming traffic of flow $f \in F$ is scheduled over BSS $b \in B_{s_f}$, then $\lambda_{f,b}^{in} = 1$, otherwise it equals 0.
- $\lambda_{f,b}^{out} \in \{0,1\}$; the path for the outgoing traffic of a flow. If the outgoing traffic of flow $f \in F$ is scheduled over BSS $b \in B_{s_f}$, then $\lambda_{f,b}^{out} = 1$, otherwise it equals 0.
- $\gamma_{s,t,b} \in \{0,1\}$; represents the connection between a station and an infrastructure device. The decision variable equals to 1, if a station $s \in S$ on technology $t \in S_t$ is part of the BSS $b \in B_s \cap B_t$, otherwise it equals 0. In other words, we assume that per technology a station can only be connected to one AP or base station.

- $\delta \in [0,1]$; models the maximal load over all BSS. In other words, this represents the load of the BSS with the highest amount of traffic assigned to it.

As an objective function, the model maximizes the total rate (bandwidth) of the network-wide traffic, both incoming and outgoing, while minimizing the relative maximal load over all BSS:

$$max(\omega \cdot (\sum_{f \in F} \tau_f^{in} + \tau_f^{out}) + (1-\omega) \cdot (-\delta) \cdot (\sum_{b \in B} \chi_b))$$

This objective function consists of two weighted subfunctions that need to be optimized (with the relative weight between them denoted by ω). The first subfunction represents the total assigned rate across all flows (which needs to be maximized). The second part represents the division of the load across all available BSSs. The idea is to minimize the maximal relative load, denoted by δ, across all BSSs [162]. As many mathematical solvers do not allow the usage of maximization or minimization functions within the objective function, δ is bounded by the final constraint. Note that the multiplication of δ with $\sum_{b \in B} \chi_b$ is only needed for normalization. While the goal is to maximize network-wide throughput, the second objective is necessary to spread all connected devices over APs and technologies. This limits the probability that the BSS becomes overloaded when a new device joins the network and connects to that BSS.

We complete the MILP formulation by defining several constraints. We first define a constraint that limits the total rate over all traffic flows on a station, going over a certain BSS, by the maximal rate supported by the configuration of that station:

- $\forall s \in S, \forall b \in B_s : \sum_{f \in F_s} \lambda_{f,b}^{in} \cdot \tau_f^{in} + \lambda_{f,b}^{out} \cdot \tau_f^{out} \leqslant d_{s_f,b} \cdot b_{s_f,b}$

Note that this constraint can be linearized by replacing the multiplication by a novel decision variable [158]. Next, we define two constraints that guarantee the conservation of flows in the network (i.e., the right endpoints):

- $\forall f \in F : \sum_{b \in B_{s_f}} \lambda_{f,b}^{in} = 1$
- $\forall f \in F : \sum_{b \in B_{s_f}} \lambda_{f,b}^{out} = 1$

Furthermore, we also need to make sure that a station can be connected to only one BSS per technology (this corresponds to reality where a device is in general only equipped with a single radio per technology):

- $\forall s \in S, \forall t \in T_s : \sum b \in B_s \cap B_t \gamma_{s,t,b} = 1$
- $\forall s \in S, \forall t \in T_s, \forall b \in B_s \cap B_t, \forall f \in F_s : \lambda_{f,b}^{in} \leqslant \gamma_{s,t,b}$
- $\forall s \in S, \forall t \in T_s, \forall b \in B_s \cap B_t, \forall f \in F_s : \lambda_{f,b}^{out} \leqslant \gamma_{s,t,b}$

Finally, we define the constraint that bounds the value of δ for balancing the load across BSSs, while also making sure that the capacity of the BSSs and their underlying technologies is not exceeded. This constraint can be linearized as mentioned previously [158].

- $\forall b \in B : \sum_{f \in F} \lambda_{f,b}^{in} \cdot \tau_f^{in} + \lambda_{f,b}^{out} \cdot \tau_f^{out} \leqslant \delta \cdot \chi_b$

5.2.3 Parameter estimation

In the next section, we explain how monitoring information is acquired from the underlying framework and fed into the MILP to calculate the optimal configuration. While some of the gathered monitoring information, like station and traffic information, can be used directly without the need for further processing, some other information is also required. A key element for determining an optimal configuration is to have an accurate overview of the available bandwidth per BSS. The big impact of determining the actual available bandwidth of wireless links on the results of load balancing approaches has been shown in literature and in the previous Chapter 4 [32]. The actual bandwidth of wireless technologies depends on several parameters such as the theoretical physical bandwidth, the configuration of APs, interference of other devices within or outside the network, and the amount of traffic in the network. Estimating each of these parameters is very challenging and is in some cases a separate research problem on its own (e.g., interference modeling). In order to avoid the use of complex and resource intensive theoretical models, we make use of the approximation function χ_b to estimate the capacity of the wireless technologies. Note that we reuse the definition as stated in Section 4.3.3. For each BSS $b \in B$, we define χ_b as follows:

$$\chi_b(\alpha, \beta) = \alpha \cdot (\sum_{f \in F} \lambda_{f,b}^{in} + \lambda_{f,b}^{out}) + \beta$$

The parameters α and β are technology dependent and capture the specifics of the wireless network under consideration. The previously described dynamic approach (cf. Section 4.3.4) allows us to capture the specific characteristics of individual heterogeneous networks. Moreover, as characteristics of the wireless environment change over time, this method can be rapidly re-executed if needed.

Furthermore, the MILP also requires the data rate (depending on the MCS) and bit error rate of the station $s \in S$ for a specific configuration, respectively denoted by $d_{s,b}$ and $b_{s,b}$. For the two parameters a mapping can be constructed: in case of the first parameter this is a mapping from measured RSSI values to MCS values (and theoretical data rate). For the second parameter $b_{s,b}$, a linear function can map the measured RSSI values to packet loss, in order to correct the theoretical achievable data rate. Both mappings can be experimentally determined by using the well-known fingerprinting approach to record MCS and packet loss values at different distances (and thus different RSSI values) in the network environment [163]. These mappings can be recreated to adjust for dynamic changes to the network environment, in a similar matter as for the α and β parameters.

5.2.4 Deployment and interaction with underlying framework

In the previous Chapter 4, and in particular in Section 4.4.2, we discussed how the proposed algorithms could be deployed on top of different network management solutions. The deployment considerations for the newly proposed load balancing MILP formulation are identical to those previously discussed. As such, the ORCHESTRA framework seems the most appropriate framework to use, due to the centralized control, monitoring features, and network management functionalities. The interaction with the framework is summarized below.

Through the framework, we interact with the network and its devices in the following ways: in a regular interval, all VMACs send monitoring information to the controller that keeps the most recent information stored. For each flow, the following information is stored: the number of transmitted and received packets, the number of transmitted and received bytes, the source, the destination, and the type. Furthermore, per link, information such as the number of packet errors, the amount of transmitted and received packets, MAC throughput, link availability, and the theoretical physical rate is reported. Finally, also information regarding the wireless technologies is stored, like the RSSI values for all APs that are in range per station, for a specific technology. The necessary information to calculate the optimal configuration (e.g., flow rates, flow destinations, and available BSSs per station) is gathered from the stored monitoring information and passed to the MILP. In turn, after the MILP has calculated the optimal configuration, we translate this configuration from the MILP variables to specific per-device VMAC rules. Finally, the controller of ORCHESTRA handles the transmission of the updated rules to each device and the configuration is thus rolled-out.

The question that remains is when exactly we have to run the load balancing algorithm. This clearly depends on the dynamic characteristics of the network and its environment as in a very static scenario it would only be a waste of resources when the algorithm is running almost continuously. But in contrast, this could be the right thing to do in a very dynamic scenario. An example of such a highly dynamic environment could be the VANETs depicted in Figure 1.2. The topology and devices in such networks are highly volatile depending on the number of cars passing by while requiring reliable real-time communication. In order to have an approach that can be utilized across a multitude of scenarios and networks, we propose to trigger the execution of the algorithm when dynamic changes to the network are detected in the monitoring information. This could, for instance, be a variation in one of the flow rates of at least x %, or when flows have joined or left the network, as discussed in more detail in Section 4.4.1. Note that while this repetitive execution allows reacting to dynamic behavior such as station mobility or changed traffic demands, this also requires a near execution time, of at most a few seconds, of the algorithm. Otherwise, the algorithm can not be used within the highly dynamic context of wireless networks and does not meet QoS requirements.

5.3 Heuristic approaches

Optimally solving the MILP problem formulation scales exponentially in terms of the number of devices and flows in the network. As such, heuristic solutions are needed for larger scenarios. To this extent, we propose two heuristic approaches in this section.

When solving the multi-technology load balancing problem addressed in this chapter, it is necessary to balance both stations across the available infrastructure devices (i.e., BSSs) and flows across different paths (i.e., connections and technologies). Both are clearly linked together as flows can only be scheduled across established paths or connections between the corresponding station and the infrastructure. However, it could be that the capacity of the technologies of the current connections is not sufficient to schedule all the flows from a certain station. This would mean that new connections need to be established to less occupied infrastructure devices, if possible. The previously introduced MILP formulation performs the station and flow load balancing jointly while finding the network configuration with the highest possible overall throughput. In order to reduce the complexity and computation time, we explore the following approach that consists of two steps. First, we load balance stations across the available infrastructure devices and resources. Second, we route the flows across the different available paths, established in the first step. Furthermore, we make use of the same inputs as the MILP formulation (defined in Section 5.2.1). Note that also the interaction with the underlying network and the deployment considerations remain the same. We propose two different heuristics that are based on this idea of splitting the overall problem.

The inputs for the models are nearly identical to those of the previously defined full problem formulation in Section 5.2.2. In other words, we reuse the network model defined in Section 5.2.1 and the linear approximation function χ_b. Additionally, we define one more input:

- $\forall s \in S, \forall t \in T_s, \forall b \in B_s \cap B_t : \exists q_{s,b,t}$; representing the quality of the connection between the station s and the BSS b using technology t. This value will vary according to the signal strength and distance between the different devices. In practice, we use the monitored RSSI values for this. Note that since RSSI values are negative, we take the negative of the recorded RSSI values when mapping them to $q_{s,b,t}$.

This additional input is required to help in the station association process, due to the clear separation in sub-problems and the fact that rates, in particular, the allowed data rate (denoted by $d_{s,b}$) for a station, can only be considered in the flow scheduling step.

5.3.1 Near-optimal two-step linear programming approach

In our first heuristic approach, we propose to sequentially solve the two sub-problems, as identified above, using linear programming. This algorithm is denoted as Split-MILP throughout the remainder of the chapter. We construct, respectively, for both sub-problems an MILP model. The inputs for the models are described above and we start by defining the formulation for the station association step.

We define the following decision variables:

- $\gamma_{s,t,b} \in \{0,1\}$; represents the connection between a station and an infrastructure device. The decision variable equals to 1, if a station $s \in S$ on technology $t \in S_t$ is part of the BSS $b \in B_s \cap B_t$, otherwise it equals 0. In other words, we assume that per technology a station can only be connected to a single AP or base station.

- $\delta \in [0, |S|]$: represents the relative maximal difference in load across all BSSs. Note that this value is bounded by the total number of stations present in the network, representing the extreme case that all stations are connected to a single BSS.

As an objective function, we define the following:

- $min \left(\left(\sum_{s \in S, t \in T_s, b \in B_s \cap B_t} \left(\frac{q_{s,b,t}}{\max\limits_{b' \in B_s} (q_{s,t,b'})} \cdot \gamma_{s,t,b} \right) \right) + \left(\frac{\delta}{|s|} \right) \right)$

This objective function consists of two parts that, respectively, minimize the RSSI values (denoted by $q_{s,t,b}$) for a connection between s and b over technology t and the load difference across the different BSSs (denoted by δ). The two fractions in the function are needed for normalizing the respective parts to the interval $[0, 1]$. In other words, the goal of this model is to assign stations to infrastructure devices (for every supported technology) that are as close as possible, while spreading to load as best as possible.

To round up the model for station assignments, we define two constraints. The first constraint calculates the difference in load (the number of connected stations) between each unique pair of two BSSs:

- $\forall b, b' \in B : |L_b - L_{b'}| \leqslant \alpha$ *with* $b \neq b'$;
  where $L_b = \forall b \in B : \sum_{s \in S} \sum_{t \in T_s} \sum_{b' \in B_s \cap B_t} \gamma_{s,t,b'}$ *with* $b = b'$

The second constraint makes sure that a station can be connected to only one BSS per technology:

- $\forall s \in S, \forall t \in T_s : \sum b \in B_s \cap B_t \gamma_{s,t,b} = 1$

Next, we define the model for the flow scheduling program. This is very similar to the formulations defined in Chapter 4, with the difference that the mobility of stations is taken into account (through the use of $d_{s,b}$ and $b_{s,b}$). Note that in this formulation we use the optimal values calculated for the decision variable $\gamma_{s,t,b}$, during the first station association step, as an input.

The following decision variables are defined:

- $\tau_f^{in} \in \left[0, r_f^{in}\right]$; the total incoming rate assigned to a flow $f \in F$.
- $\tau_f^{out} \in \left[0, r_f^{out}\right]$; the total outgoing rate assigned to a flow $f \in F$.
- $\lambda_{f,b}^{in} \in \{0, 1\}$; the path for the incoming traffic of a flow. If the incoming traffic of flow $f \in F$ is scheduled over BSS $b \in B_{s_f}$, then $\lambda_{f,b}^{in} = 1$, otherwise it equals 0.
- $\lambda_{f,b}^{out} \in \{0, 1\}$; the path for the outgoing traffic of a flow. If the outgoing traffic of flow $f \in F$ is scheduled over BSS $b \in B_{s_f}$, then $\lambda_{f,b}^{out} = 1$, otherwise it equals 0.

As an objective function, the model maximizes the total rate (bandwidth) of the traffic, both incoming and outgoing, across all the traffic flows:

- $max \sum_{f \in F} \tau_f^{in} + \tau_f^{out}$

To conclude the model, we define several constraints. The first constraint guarantees that the capacity of BSSs and their underlying technologies is not exceeded:

- $\forall b \in B : \sum_{f \in F} \lambda_{f,b}^{in} \cdot \tau_f^{in} + \lambda_{f,b}^{out} \cdot \tau_f^{out} \leqslant \chi_b$

Next, we define a constraint that limits the total rate over all traffic flows on a station, going over a certain BSS, by the maximal rate supported by the configuration of that station:

- $\forall s \in S, \forall b \in B_s : \sum_{f \in F_s} \lambda_{f,b}^{in} \cdot \tau_f^{in} + \lambda_{f,b}^{out} \cdot \tau_f^{out} \leqslant d_{s_f,b} \cdot b_{s_f,b}$

Furthermore, we define two constraints that guarantee the conservation of flows in the network (i.e., the right endpoints):

- $\forall f \in F : \sum_{b \in B_{s_f}} \lambda_{f,b}^{in} = 1$
- $\forall f \in F : \sum_{b \in B_{s_f}} \lambda_{f,b}^{out} = 1$

Finally, we make sure that flows are only scheduled across existing connections, based on the optimal configuration from the station association:

- $\forall s \in S, \forall t \in T_s, \forall b \in B_s \cap B_t, \forall f \in F_s : \lambda_{f,b}^{in} \leqslant \gamma_{s,t,b}$
- $\forall s \in S, \forall t \in T_s, \forall b \in B_s \cap B_t, \forall f \in F_s : \lambda_{f,b}^{out} \leqslant \gamma_{s,t,b}$

5.3.2 Greedy heuristic

In contrast to our first heuristic (Split-MILP) that is based on linear programming, we now present a more conventional greedy approach. The overall idea is to sequentially loop over all stations and assign them (per technology) one by one to the most appropriate infrastructure device. The latter is determined by combining

Algorithm 1 First step: Station Association.

```
1: for $s \in S$ do
2:     for $t \in T_s$ do
3:         Let $W[1 \ldots |b|]$ be a new array                    $\triangleright b \in B_s \cap B_t$
4:         for $b \in B_s \cap B_t$ do
5:             if $\max_{b' \in B} \sum_{s' \in S} \gamma_{s,t,b} > 0$ then
6:                 $W[b] \leftarrow \frac{rssi_{s,b,t}}{\max_{b' \in B_s} rssi_{s,b',t}} + \frac{\sum_{s' \in S} \gamma_{s,t,b}}{\max_{b' \in B} \sum_{s' \in S} \gamma_{s,t,b}}$
7:             else
8:                 $W[b] \leftarrow \frac{rssi_{s,b,t}}{\max_{b' \in B_s} rssi_{s,b',t}}$
9:             end if
10:        end for
11:        $\gamma_{s,t,b} \leftarrow 1,$ with $b \in B_s$ and $W[b] = \min_{b' \in B_s} W[b']$
12:    end for
13: end for
```

the distance and load values. In the second phase, we perform the same identical procedure for scheduling flows to the connections with the most remaining capacity. Note that all flows are scheduled, even if no unsaturated connections exist anymore. In such a case, it is up to the underlying transport protocols to adjust the rates of the different traffic flows.

In the first step, depicted in Algorithm 1, we start by iterating over all stations in S. This list of stations can be sorted based on a number of criteria. For instance, according to the arrival of the stations in the network or on the decreasing sum of rates (across all flows) per station. We have opted for the first more neutral option, where stations with a large amount of traffic are not favored over others. For each supported technology per station, we create a map with an assigned score per available BSS (line 3). This score combines the relative distance from the station to each infrastructure device with the load on that AP or base station (lines 4-10). This score allows us to take into account the mobility of stations and the shared spectrum per infrastructure device. We distinguish two cases. First, the most common case where already at least one station has been assigned to a BSS, meaning the max load across all BSSs is larger than zero (line 6). Second, the initial case where no load was assigned yet (line 8). Here, we only take into account the relative distance to avoid a division by zero. Next, the station is assigned to the BSS with the lowest score (line 11).

The second step of the heuristic is shown in Algorithm 2. We first create an array where we store the remaining capacity (initially the max capacity) per BSS (line 1 and 2). A second array represents the total assigned rates per BSS (line 3). We then iterate over all flows in F. Once again these flows can potentially be sorted based on their rates. However, this could potentially benefit larger flows (also

Algorithm 2 Second step: Flow Scheduling.

```
1: Let $C[1 \ldots |B|]$ be a new array
2: $C[b] \leftarrow \chi_b$                                        $\triangleright \forall b \in B$
3: Let $T[1 \ldots |B|]$ be a new array
4: for $f \in F$ do
5:     $\lambda^{in}_{f,b} \leftarrow 1$ with $\gamma_{s_f,t,b} = 1$ and $C[b] = \max\limits_{\gamma_{s_f,t,b'}=1} C[b']$    $\triangleright$ $\forall b' \in B_{s_f}, \forall t \in T$
6:     $T[b] \leftarrow T[b] + \min\left(\left(d_{s_f,b} \cdot b_{s_f,b}\right), r^{in}_f\right)$
7:     $C[b] \leftarrow \max\left(0, (\chi_b - T[b])\right)$
8:     $\lambda^{out}_{f,b} \leftarrow 1$ with $\gamma_{s_f,t,b} = 1$ and $C[b] = \max\limits_{\gamma_{s_f,t,b'}=1} C[b']$    $\triangleright$ $\forall b' \in B_{s_f}, \forall t \in T$
9:     $T[b] \leftarrow T[b] + \min\left(\left(d_{s_f,b} \cdot b_{s_f,b}\right), r^{out}_f\right)$
10:    $C[b] \leftarrow \max\left(0, (\chi_b - T[b])\right)$
11: end for
```

denoted as elephant flows in some research areas). As such, we use the arbitrarily order in which flows arrive in the network. For each flow, we first assign a path for the incoming traffic by selecting the BSS with the most capacity remaining (line 5). Next, we update the traffic assigned to the selected BSS by adding the minimum from the allowed rate on the station (depending on the MCS) and the incoming rate of the flow (line 6). On line 7, we also update the remaining capacity of the selected BSS by subtracting the assigned rates from the approximation function χ_b. By doing so, we account for the loss in maximal capacity of a wireless technology when more and more devices are added. After the selection of the path for the incoming traffic, we repeat the same for the outgoing traffic on lines eight to ten. This procedure is repeated until all flows are scheduled over a certain path.

5.4 Evaluation and discussion

In this section, we evaluate the presented load balancing approach across a variety of scenarios. We focus on comparing the performance of the complete MILP algorithm against the two heuristics and demonstrating the scalability and versatility of the proposed load balancing approach. For this, we mainly make use of simulation results obtained from the NS-3 event-based network simulator, complemented with a direct algorithmic evaluation in Python. The structure of this section is as follows. First, we discuss the evaluation setup and the topology of the different scenarios. Note that this setup is similar to the one used for the evaluation in Chapter 4. Next, we discuss in detail how we selected the values for the different parameters. Afterwards, the performance of the approach, in terms of achieved throughput and execution time, is evaluated across a variety of scenarios.

5.4.1 Evaluation setup

Most simulations are conducted using the NS-3.27 network simulator, where we implemented the entire ORCHESTRA framework, the MILP problem formulation, and the two heuristic approaches [160]. To optimally solve the MILP and the Split-MILP formulations, we make use of the Gurobi Optimizer (7.5.2) [156]. All experiments are conducted on a single core of an Intel® Xeon® E5-2680 Processor running at 2.8 GHz and with 8 GB RAM. Furthermore, we also extended the basic NS-3.27 implementation to allow for multi-channel Wi-Fi networks. During all of our experiments, we assume that two Wi-Fi technologies are present using, respectively, 2.4 GHz and 5 GHz frequency bands. Note that as our load balancing approach is fully technology independent, it is of less importance which technologies are selected for the evaluation. Every scenario has at least two APs that support both technologies. Dynamic rate adaptation for all devices is made possible through the Minstrel rate adaptation algorithm. Besides the generated traffic flows themselves, also the management traffic is considered in the simulations. In other words, the packets that contain monitoring information and configuration instructions, sent between the devices and the controller or vice versa, are also generated and transmitted. As such, our results consider the overhead of the management interactions.

As NS-3 emulates all packets within a network, simulation time grows rapidly when increasing the network size and traffic amount. In order to allow us to investigate the scalability of the approach to larger networks and to perform a rapid evaluation of algorithm parameters, we created a second experimental setup, outside of the NS-3 simulator. In Python, we implemented, on a 2016 Intel NUC, both the MILP and the two heuristic approaches. As before, the Gurobi Optimizer (7.5.2) is used to solve the MILP and Split-MILP algorithms. Furthermore, we created a framework that could artificially generate the required inputs for the algorithms. This allows us to easily test the impact of varying configurations of stations, APs, and flows, without the need for any network interaction or full network simulation. This was mainly used to investigate the execution time and scalability of the approaches, and not for the analysis of optimality or network performance. For each experiment, we will clearly specify the ranges of values that were evaluated.

For every scenario throughout the evaluation, we provide a comparison to a fully distributed baseline, where each device decides for itself to which AP to connect, based on the best RSSI values. Furthermore, we assume that when the RSSI of the current connection drops below a certain threshold, a better connection is selected (if present) for that device on that specific technology. The selection of the threshold value will be discussed in the next section. In other words, the baseline corresponds to the current state-of-the-art, where an arbitrary multi-technology management solution is in place (either the VMAC from ORCHESTRA or one of the ones discussed in Section 2.2), without the centralized intelligence, but with seamless handovers. Furthermore, we also compare against the performance of a fully randomized algorithm that selects uniformly at random for each station the corresponding infrastructure device (i.e., BSS) to connect to, and for each incom-

Device type	**Rate boundaries per flow type**		
(and mobility)	**Download**	**Video stream**	**Conference call**
Laptop (mobile)	10–30 Mbps	8–20 Mbps	4–10 Mbps
HD Television	5–25 Mbps	10–20 Mbps	5–10 Mbps
4K Television	5–25 Mbps	15–25 Mbps	7.5–12.5 Mbps
Tablet (mobile)	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps
Smartphone (mobile)	1–8 Mbps	2.4–9 Mbps	1.2–4.5 Mbps

Table 5.1: Overview of the devices, and the supported flow rates, used in the scenarios.

ing and outgoing flow its path.

In order to generate representative network topologies and conditions, several types of devices are defined, each with different mobility characteristics and traffic rates. This information is depicted in Table 5.1. The exact number of devices, the assigned flow type, and the rate of the flow are chosen uniformly at random between an upper and lower bound, based on the involved device and depending on the scenario. Each mobile device (all devices except for the televisions) moves around according to the Random Waypoint Model within a certain area, with a random start position and a uniformly random chosen speed between 0.3-0.7 $\frac{m}{s}$. The size of the area and the wait times at the waypoints depend on the scenario. Moreover, in the static scenarios, the flow rates do not change over time, while in the other scenarios the download flows will consume as much bandwidth as possible (reflecting their actual behavior) and traffic flows arrive and leave the network dynamically. Assuming a static flow rate for the first part of the evaluations allows us to better estimate the impact of only the mobility aspect and to verify the impact of using monitored flow rates. The size of the download is uniformly at random chosen between 10 MB and 10 GB. We assign one flow per device and as such do not assume the concurrent usage of both Wi-Fi interfaces, as this is generally not supported by current hardware. Note, that the flow rates were selected based on representative figures from literature of existing applications [161]. We decided to use only TCP traffic flows, as current Internet traffic is dominated by TCP [87]. Finally, for every described scenario, results are averaged over 20 different randomly generated flow and topology configurations.

5.4.2 Selection of parameters

Both in the algorithms themselves as in the interaction with the network there are a number of parameters that can potentially have a large impact on the evaluation results. Below we discuss all parameters one by one and clearly highlight how the values are selected.

- Weight w for MILP objective function: as the objective function of the MILP is built out of two sub-functions, respectively, the throughput maximization function and the load balancing function, a weight is needed to combine both goals. Using our Python experimental setup, we optimally

solved the MILP for a large number of scenarios, testing out a range of weights per scenario. We use Gurobi to calculate the optimal configuration for a specific scenario and weight. As such, we perform a parameter sweep to determine the best value of w. In order to determine the best value of w, we calculate for every scenario a score. This score is based on two things. First the objective score calculated by Gurobi for that specific scenario and weight. This objective score indicates the amount of achieved network-wide throughput. Second, we generate a random number (uniformly distributed) of additional flows with certain rate requirements. We try to arbitrarily assign these flows to the previously calculated configuration. The amount of capacity requested by the flows that is available in the network (normalized over the total capacity of the network), is added to the previously calculated objective score for that specific scenario and weight. In case no capacity is available for the additional flows, the objective score is not altered. In this manner, we obtain a score that takes into account the existing throughput requirements of the network and the robustness towards additional traffic in the nearby future. Note that the latter is the main reason behind the load balancing requirement.
Per specific combination of scenario and weight, we repeated this 20 times. In total we varied the number of stations between 5 and 15, the number of APs between 2 and 5, and the number of additional flows between 1 and half of the number of stations selected. While considering weights between 0 and 1 with a step-size of 0.01 between the different considered values. Finally, we averaged results per weight across all scenarios and normalized the scores between 0 and 1.
Results are depicted in Figure 5.1. The figure shows for each weight (averaged over the 20 runs), the calculated score, the network-wide throughput achieved by the MILP algorithm, and the maximal difference in load between two BSSs (denoted as MILP load difference). From the figure, it is clear that the best performance was achieved when using a weight of 0.91. Furthermore, we see that for the value of 0.91, the network-wide throughput (as calculated by the MILP) is at its highest, while the difference in load between the different BSSs is not at its lowest, but also not at its highest. Intuitively, this clearly shows that the major objective is to maximize network-wide throughput. However, to account for the dynamic behavior of traffic, the weight needs to be selected where the difference in load is minimized as much as possible, without strongly impacting network-wide throughput. During all the following experiments, the weight of 0.91 will be used. Note that for visualization reasons, we do not show the weights below 0.5, as they scored the lowest of all, while also the MILP throughput is normalized between 0 and 1.

- Parameters α and β for the χ_b capacity approximation function: here we applied the method as described in Section 5.2.3. Per technology, we considered a number of stations between 1 and 15, while varying the flow rates

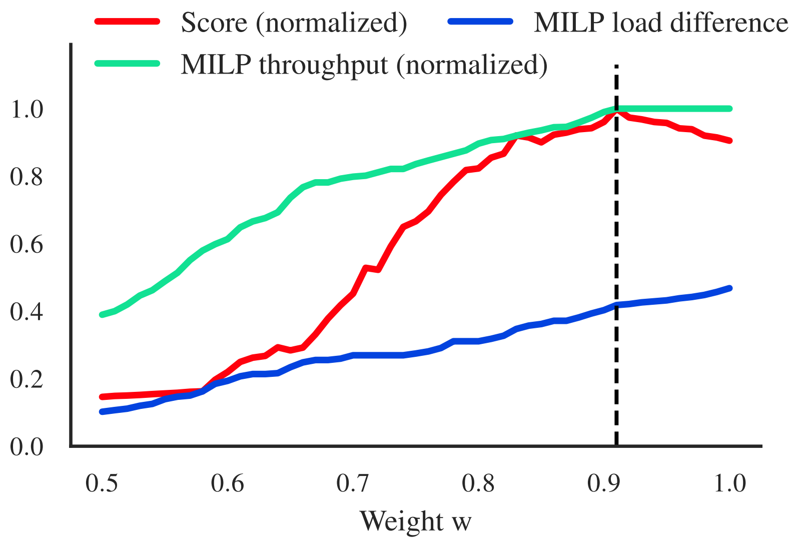


Figure 5.1: Normalized score, network-wide throughput (normalized), and maximal load difference over different scenarios for different values of the weight in the MILP objective function.

between the theoretical data rate of the particular technology and 1 Mbps. We determined the following parameters: for the function χ_b, α and β are respectively, for 2.4 GHz Wi-Fi -1.74 and 57.58, and for 5 GHz Wi-Fi -3.21 and 112.99.

- Algorithm execution parameters: the execution of the algorithm (either the MILP or one of heuristics) is triggered by the real-time monitoring component when dynamic changes to the network have been detected (e.g., a variation in one of the flow rates of at least 25 %, or the arrival of a new flow) or when it has been 10 s since the last execution. The latter ensures that the network configuration is optimized on a regular base, even in very static environments. The value of the parameter can be chosen based on the applicable environment. The first value (of 25 %) was chosen based on a similar experiment as conducted for the weight of the objective function. In our NS-3 implementation, we tried out different values and selected the one with the highest impact. Furthermore, to avoid oscillations in the decision-making and allow changes to occur in the network, there should be at least 2 s between two consecutive executions. The other two values were selected based on expert knowledge.

- MILP time limit: to ensure the continuation of experiments and thus ending simulations in a feasible amount of time, a time limit is set for solving the

	Number of devices			
Device type	**Home** (20x10 m)	**Small office** (25x10 m)	**Large office** (30x15 m)	**Flow types**
APs	2	3	4	N/A
Laptop	2	9	12	Download/Conf. call
HD TV	0	1	1	Video stream
4k TV	1	0	1	Video stream
Tablet	2	1	2	All types of flows
Smartphone	3	5	8	All types of flows
Total	10	19	28	

Table 5.2: Setup for static scenarios.

MILP. Here, a value of 900 s was chosen. Note that this value was chosen a magnitude larger than the amount of time maximally available between two executions and required for reactive real-time optimizations. This allows us to sufficiently investigate the scalability of the MILP in terms of execution time and show that it is not feasible to solve the MILP in real-time.

- Baseline RSSI threshold: as mentioned in the previous section, a threshold is used to determine when a device needs to handover to a better connection (if existing). We chose a threshold of -75 for this, as this value is considered to still correspond to an average connection quality. Note that during some of the following experiments, we also tried out other threshold values (e.g., -65, -70, and -80), but this only led to limited differences.

5.4.3 Static flow rate scenarios

In order to get a first impression of the performance of the different approaches, we created three basic scenarios with varying topologies. As depicted in Table 5.2, these scenarios grow in size and density. The results for all three scenarios are shown in, respectively, Figures 5.2, 5.3, and 5.4. The graphs compare the baseline, random algorithm, MILP formulation, and the two heuristics (respectively, denoted as split-MILP and greedy heuristic) to the sum of the desired flow rates. The latter is known as fixed flow rates are used here. Across all graphs, we clearly see a significant improvement by our presented algorithms in comparison to the distributed baseline and to the random algorithm. Moreover, we see nearly no differences, in terms of achieved network-wide throughput, between the heuristics and the optimal MILP approach.

For the home scenario, we can report the following rates ($\pm$ the standard error), respectively, for the baseline, random algorithm, MILP, split-MILP, and the greedy heuristic: 81.61 Mbps ($\pm$2.62), 87.34 Mbps ($\pm$2.24), 90.32 Mbps ($\pm$2.34), 90.15 Mbps ($\pm$2.31), and 89.96 Mbps ($\pm$2.38). As such, there is an improvement of, respectively, 10.67, 10.40 and 10.16 % compared to the baseline for the optimal

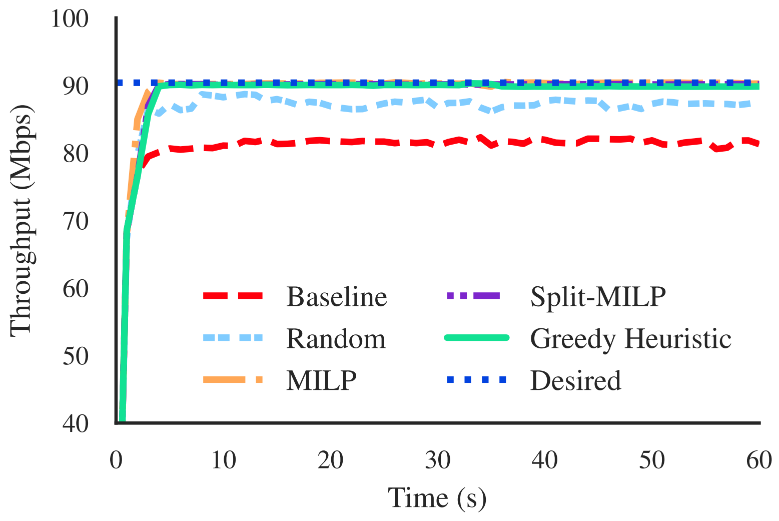


Figure 5.2: Throughput as a function of time for the home scenario, comparing the MILP formulation, the two heuristics, the random algorithm, and the baseline.

and heuristic solutions. Similarly, there is an increase of, respectively, 3.41, 3.22, and 2.86 % towards the random algorithm. Note that the random algorithm performs better than the baseline due to the fact that by selecting connections and flow routes at random, a simple form of load balancing is performed (on average). Furthermore, the differences of 0.17 Mbps between the MILP and the split-MILP and of 0.36 Mbps between the MILP and the greedy heuristic, are negligible. As the total desired rate is 90.40 Mbps (±2.35), it is clear that our approach succeeds in providing the optimal network configuration. Similarly for the small office scenario, the following average network-wide rates are achieved: 131.46 Mbps (±3.73), 179.85 Mbps (±3.52), 193.90 Mbps (±3.76), 190.40 Mbps (±3.31), 192.63 Mbps (±3.14), for respectively, the baseline, random algorithm, MILP, split-MILP, and greedy heuristic. The increases towards the baseline are larger than for the home scenario: 47.50, 44.83, and 46.53 %, while also the increase towards the random algorithm is larger (respectively, 7.81, 5.87, and 7.11 %). The difference between the two heuristics and the optimal MILP solution are, respectively, 3.50 and 1.27 Mbps, which is once again negligible. The same can be said for meeting the requirements of the flows as the total desired rate is 195.21 Mbps (±3.46). For the large office scenario, the largest network considered, it was impossible to calculate solutions for the MILP within the time limit of 900 s. For, respectively, the baseline, random algorithm, and the two heuristics the following rates are achieved: 179.71 Mbps (±3.61), 239.41 Mbps (±3.41), 270.25 Mbps (±3.36), and 283.60 Mbps (±3.31). This means that there is an increase 90.54 Mbps or

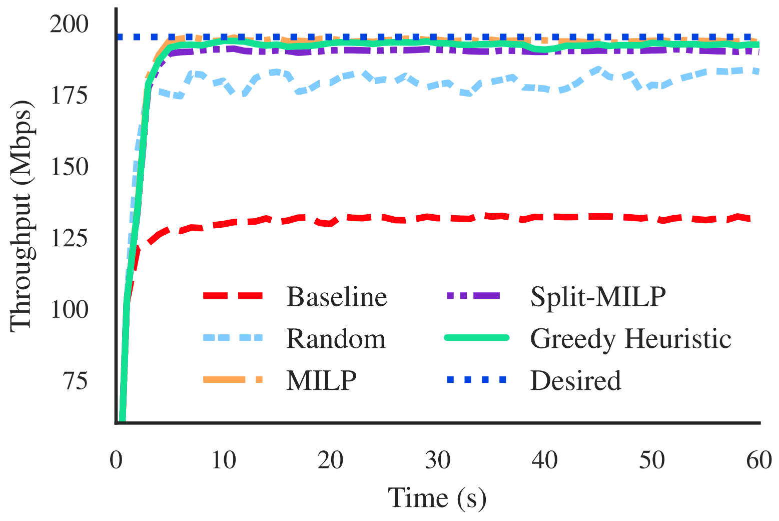


Figure 5.3: Throughput as a function of time for the small office scenario, comparing the MILP formulation, the two heuristics, the random algorithm, and the baseline.

50.38 % for the split-MILP in comparison the to the baseline. For the greedy heuristic this increase is 103.89 Mbps or 57.81 %. Compared to the random algorithm, there is an increase of 12.88 % for the split-MILP and of 18.46 % for the greedy approach. If we compare the throughput of the heuristics to the overall desired rate, we see that the split-MILP is 28.76 Mbps off, while the greedy heuristic is 15.41 Mbps off. This difference towards the desired rate is a bit higher than in the other two cases. The reason for this is that the limits of the wireless technologies are being reached. This is also the reason for the fluctuations that can be seen in the throughput of the heuristic in Figure 5.4.

While it is clear from the above discussion that both heuristics offer a performance that is relatively close to the optimal MILP solution (in terms of throughput), a significant difference can be noticed when comparing the two heuristics against each other. This is especially the case for the two largest scenarios. For the home scenario, the greedy heuristic is barely outperformed by the split-MILP. However, in the case of the small office scenario the split-MILP achieves only an average throughput of 190.40 Mbps (±3.31), in contrast to the 192.63 Mbps (±3.14) that is achieved by the greedy heuristic. This difference grows in the large office scenario to, respectively, 270.25 Mbps (±3.36), and 283.60 Mbps (±3.31). This is unexpected as both heuristics exploit the same principle, while the split-MILP solves the sub-problems optimally. However, upon investigating we noticed that the underperformance of the split-MILP is caused by the fact that the real flow rates are unknown and the real-time monitoring information is used. If we would

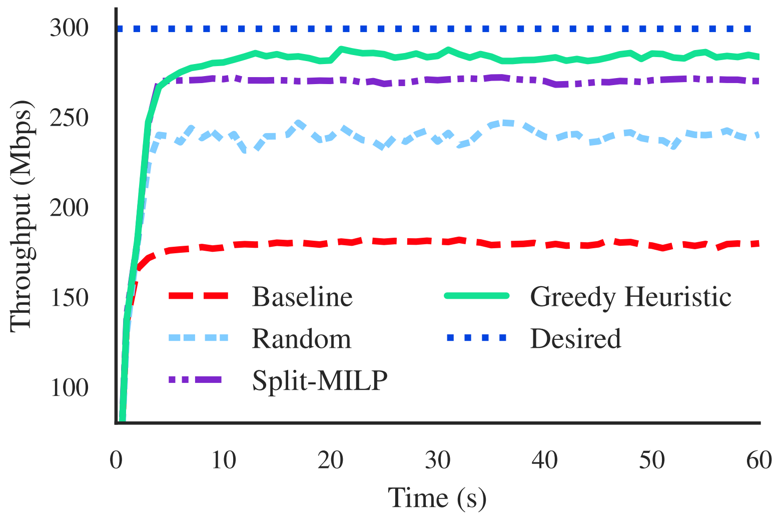


Figure 5.4: Throughput as a function of time for the large office scenario, comparing the two heuristics, the random algorithm, and the baseline.

use the desired flow rates: the following results would be achieved for, respectively, the small and large office scenario: 193.12 Mbps (±3.92), and 287.41 Mbps (±3.61). These results are slightly above the ones achieved by the greedy heuristic, as expected. Note that the other algorithms, both the optimal MILP and the greedy heuristic, do not experience this issue when using the real-time rates of the traffic flows and the improvement of using the desired flow rates is comparable to the results reported in Section 4.5.3. For instance, for the greedy algorithm in the small office scenario, a throughput of 192.81 Mbps (±3.35) is achieved when using the desired flow rates. This is in contrast to the 192.63 Mbps (±3.14), reported earlier when using the real-time monitoring information.

As already mentioned, it proved to be infeasible to optimally solve the MILP for the large office scenario. While it was possible to find a solution for the first two scenarios, the execution time was high: respectively, 16.38 s (±4.28) and 736.58 s (±39.71). Note that these execution times are significantly above the minimal interval (of 2 s) between two consecutive runs of the algorithm. Luckily, the two heuristics perform significantly better in terms of execution time. For the split-MILP, we report the following execution times, respectively, for the three scenarios: 1.12×10^{-2} s ($\pm 8.48 \times 10^{-3}$), 4.29×10^{-2} s ($\pm 2.44 \times 10^{-2}$), and 8.86×10^{-2} s ($\pm 3.24 \times 10^{-2}$). For the greedy heuristic, the following solve times apply: 8.23×10^{-5} s ($\pm 3.92 \times 10^{-5}$), 1.93×10^{-4}s ($\pm 1.24 \times 10^{-5}$), and 5.23×10^{-4}s ($\pm 2.58 \times 10^{-5}$). When comparing the execution times for both heuristics, we notice that the times of the greedy heuristic are a magnitude lower than those of the split-MILP. We will discuss the scalability of both the MILP and the two heuristic approaches in more detail in the next section but the trends are already clearly illustrated.

Finally, we consider the impact of mobility on the overall throughput. Therefore, we varied the waypoint wait times for all scenarios by additional experiments for times between 0-10 s and 10-20 s. The results, listed in Table 5.3, show that the algorithms consistently and significantly outperform the baseline and the random algorithm. Note that for the case with the highest mobility (and lowest wait times), the baseline performs significantly better, than in the other cases. We believe this to be due to the higher number of handovers, triggered by the mobility and its more reactive nature. Furthermore, the split-MILP heuristic continues to underperform for larger scenarios, when using the real-time monitoring, in comparison to the greedy heuristic.

5.4.4 Impact of network load and scalability

To investigate the scalability of the different algorithms in terms of traffic and execution time, the following scenario was created: a set of devices was randomly generated, each with a uniform randomly assigned flow with a randomly chosen type and rate. The total desired rate of all the generated flows equals a certain percentage of the total theoretical network capacity. Experiments were performed for loads of 10, 20, 30, 40, 50, 60, and 70 % of the theoretical network capacity. Moreover, the presence of 3 APs was assumed in a space of 20 by 15 m with a waypoint wait time of 5-15 s.

From Figure 5.5 it is clear that our heuristic approaches offer a significant improvement compared to the baseline. This improvement grows when the percentage of network traffic increases. Furthermore, the figure also shows the increasing difference in performance between the two heuristics for the larger scenarios. For instance, for a load of 60 % there is an increase from 135.57 Mbps ($\pm$2.50) for the baseline and 226.45 Mbps($\pm$1.79) for the random approach, to 252.76 Mbps ($\pm$3.68) and 267.60 Mbps ($\pm$3.04) for, respectively, the split-MILP and the greedy heuristic. This is an increase of, respectively, 86.44 % and 97.39 %, towards the baseline. Compared to the random algorithm, this is an increase of, respectively, 11.62 % and 18.17 %. As mentioned in the previous section, the performance of the split-MILP drops upon increasing the size of the scenarios due to the higher susceptibility towards the usage of real-time monitoring information (i.e., flow rates). If the known rates are used, a throughput of 269.03 Mbps ($\pm$2.89) is achieved by the split-MILP for a network load of 60 %

More importantly, we see that the proposed algorithms (in particular the greedy heuristic) allow, in general, to satisfy the traffic demands of all flows. Only at 60 % and 70 %, there is a difference of, respectively, 35.58 Mbps and 61.35 Mbps between the desired rates and the achieved throughput. However, this is largely due to reaching the limits of the wireless technologies as our network loads are based on the theoretical capacities, which can not be met in reality due to capacity loss at the higher layers (e.g., back-off timers, retransmissions, etc). Furthermore, we

Scenario	**Wait times**	**Baseline**	**Random**	**MILP**	**Split-MILP**	**Greedy Heuristic**
Home	0-10 s	83.16 Mbps (±3.31)	86.94 Mbps (±2.20)	89.81 Mbps (±2.35)	89.76 Mbps (±2.15)	89.74 Mbps (±2.31)
	5-15 s	81.61 Mbps (±2.62)	87.34 Mbps (±2.24)	90.32 Mbps (±2.34)	90.15 Mbps (±2.31)	89.96 Mbps (±2.37)
	10-20 s	80.32 Mbps (±2.88)	87.85 Mbps (±2.30)	90.24 Mbps (±2.29)	89.58 Mbps (±2.30)	89.35 Mbps (±2.25)
Small Office	0-10 s	157.19 Mbps (±4.70)	178.06 Mbps (±3.65)	193.23 Mbps (±3.57)	189.64 Mbps (±3.23)	191.03 Mbps (±3.80)
	5-15 s	131.46 Mbps (±3.73)	179.85 Mbps (±3.52)	193.90 Mbps (±3.76)	190.40 Mbps (±3.31)	192.63 Mbps (±3.14)
	10-20 s	135.46 Mbps (±3.98)	179.89 Mbps (±3.95)	194.58 Mbps (±3.76)	191.56 Mbps (±3.27)	193.16 Mbps (±3.19)
Large Office	0-10 s	229.47 Mbps (±6.22)	238.78 Mbps (±3.67)	N/A	269.17 Mbps (±3.48)	282.91 Mbps (±3.62)
	5-15 s	179.71 Mbps (±3.61)	239.41 Mbps (±3.41)	N/A	270.25 Mbps (±3.36)	283.60 Mbps (±3.61)
	10-20 s	178.73 Mbps (±5.06)	239.27 Mbps (±3.93)	N/A	274.82 Mbps (±3.13)	286.56 Mbps (±3.30)

Table 5.3: Impact of mobility on throughput.

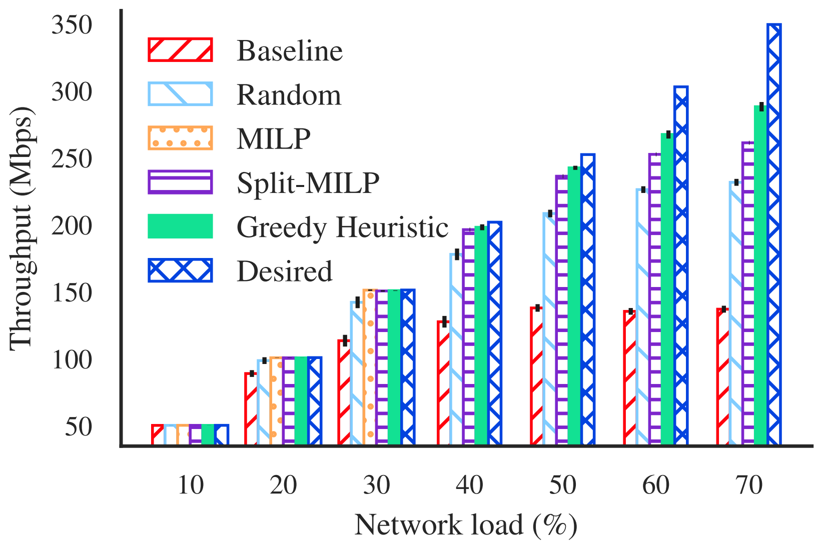


Figure 5.5: Throughput as a function of network load, error bars depict the standard error.

can also point out that there is nearly no difference between the optimal (MILP) solution and the heuristic approaches, in terms of achieved network-wide throughput. For instance, at a load of 30 % there is only a difference of 0.37 Mbps and 0.71 Mbps between the results of the optimal solutions and, respectively, the split-MILP and greedy heuristic. Note that there are no throughput results depicted for the MILP formulation for the network loads of 40 % and more due to the high computation time. This is similar to the large office scenario in the previous section.

We measured for all three algorithms the time that it takes to calculate a solution. Table 5.4 shows the averages of the measured values across the different network loads. It is clear that the computation time for the MILP scales exponentially. For instance, for only 14 flows (i.e., load of 30 %) it takes already 478.36 s (±36.39) to compute the configuration. For higher loads, it was infeasible to calculate a solution within the time limit of 900 s. This clearly indicates that the MILP solution cannot be used in very dynamic real-life wireless networks. In contrast, the computation times reported when using the heuristics are drastically lower. For instance, for the same scenario with a network load of 30 % and 14 flows, it takes only 3.39×10^{-2} s ($\pm 1.07 \times 10^{-2}$) and 1.85×10^{-4} s for, respectively, the split-MILP and greedy heuristics to find a solution. Moreover, for the scenarios with higher amounts of network load, it remains feasible for both heuristics to provide a solution in a real-time fashion. However, when comparing the two heuristics throughout the different scenarios, we note that the difference in execution time

Network Load	**# Flows**	**Execution time MILP**	**Execution time split-MILP**	**Execution time greedy heuristic**
10	6	8.17 s (± 1.08)	8.35×10^{-3} s ($\pm 2.22 \times 10^{-3}$)	4.17×10^{-5} s ($\pm 1.62 \times 10^{-5}$)
20	10	29.75 s (± 6.84)	1.65×10^{-2} s ($\pm 8.15 \times 10^{-3}$)	1.25×10^{-4} s ($\pm 7.21 \times 10^{-5}$)
30	14	478.36 s (± 36.39)	3.39×10^{-2} s ($\pm 1.07 \times 10^{-2}$)	1.85×10^{-4} s ($\pm 8.81 \times 10^{-5}$)
40	19	N/A	5.14×10^{-2} s ($\pm 2.32 \times 10^{-2}$)	2.16×10^{-4} s ($\pm 1.81 \times 10^{-4}$)
50	24	N/A	8.92×10^{-2} s ($\pm 3.30 \times 10^{-2}$)	3.29×10^{-4} s ($\pm 2.06 \times 10^{-4}$)
60	29	N/A	1.19×10^{-1} s ($\pm 5.59 \times 10^{-2}$)	4.90×10^{-4} s ($\pm 2.26 \times 10^{-4}$)
70	34	N/A	1.46×10^{-1} s ($\pm 7.13 \times 10^{-2}$)	5.11×10^{-4} s ($\pm 2.22 \times 10^{-4}$)

Table 5.4: Comparison of the execution time for the MILP and heuristic solutions, under increasing network load.

grows. For instance, for the case with a network load of 70 % and 34 flows, it already takes the split-MILP 1.46×10^{-1} s ($\pm 7.13 \times 10^{-2}$) to provide a solution, while the greedy heuristic does provide a (better) solution in 5.11×10^{-4} s ($\pm 2.22 \times 10^{-4}$).

To further investigate the scalability of the heuristic approaches, we performed a separate experiment emulating larger networks. For this, we employed the Python-based setup, as discussed in Section 5.4.1. We artificially provide the necessary inputs to the heuristic, thereby varying the number of stations and APs, while assuming the presence of 4 technologies. Depending on the heuristic, different ranges of values are used. For the split-MILP, we vary the number of stations between 10 and 300, and the number of APs between 2 and 50. On the other hand, for the greedy heuristic, we vary the number of stations between 100 and 10000 and the number of APs between 10 and 1000. For each pair of the number of stations and APs, we take the average of 20 executions, each with a randomly generated topology. Figures 5.6 and 5.7 show the resulting heatmaps, where every colored cell indicates the average time to solve the respective heuristic for a specific pair. First of all, we can see that the execution time for both algorithms mainly depends on the number of stations (and thus flows), and less on the number of APs (and their BSSs). Furthermore, we can clearly see that the greedy heuristic scales to larger topologies than the split-MILP approach. While the execution time increases when

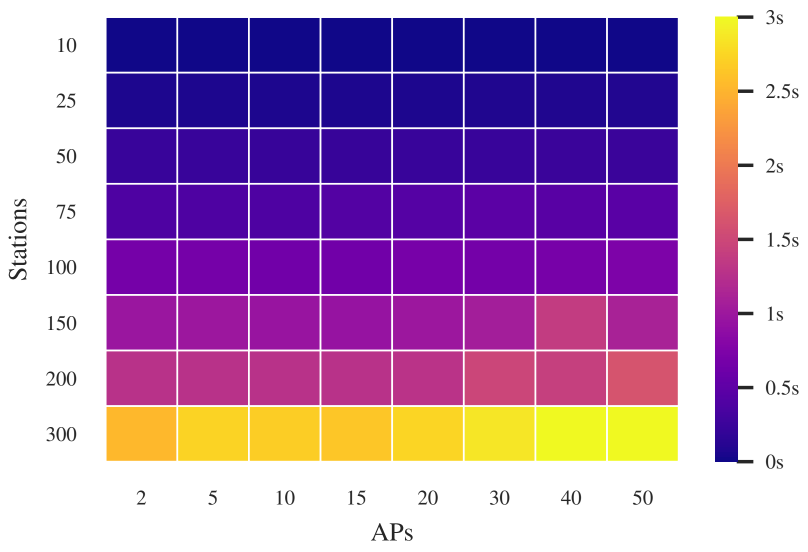


Figure 5.6: Scalability of the split-MILP heuristic in terms of stations and APs.

the number of stations grows up to 10000, it stays under 3 s for all configurations when using the greedy heuristic. In particular, it takes on average 2.95 s ($\pm$ 0.07) to calculate a solution for the largest configuration. In contrast, the execution time of the split-MILP scales already up to roughly 3 s for a topology with 300 stations. As such, this means that the greedy heuristic algorithm allows the best to react to dynamic network changes and allows us to perform optimizations in real-time, even for very large networks.

5.4.5 Dynamic flow rate scenarios

Up to now, we have only considered scenarios with static flow rates and arrival times, as this helped in determining the impact of mobility and the use of real-time monitoring. We will now consider a more dynamic scenario, as this is more realistic and the adaptability to dynamic conditions is also key for our approach. All download flows thus act as in reality and consume as much bandwidth as possible until the desired amount of data has been downloaded (or the maximum flow length has been reached). Moreover, flows arrive according to a Poisson distribution and the flow length is uniformly at randomly chosen between 5 and 15 s and 10 and 30 s, respectively for the first and second scenario. We evaluate the impact of different values for the Poisson arrival rate (λ). Furthermore, given the results from the previous sections that showed that the greedy heuristic is the best performing heuristic (in terms of achieved network-wide throughput and execution time), we only evaluate this particular algorithm.

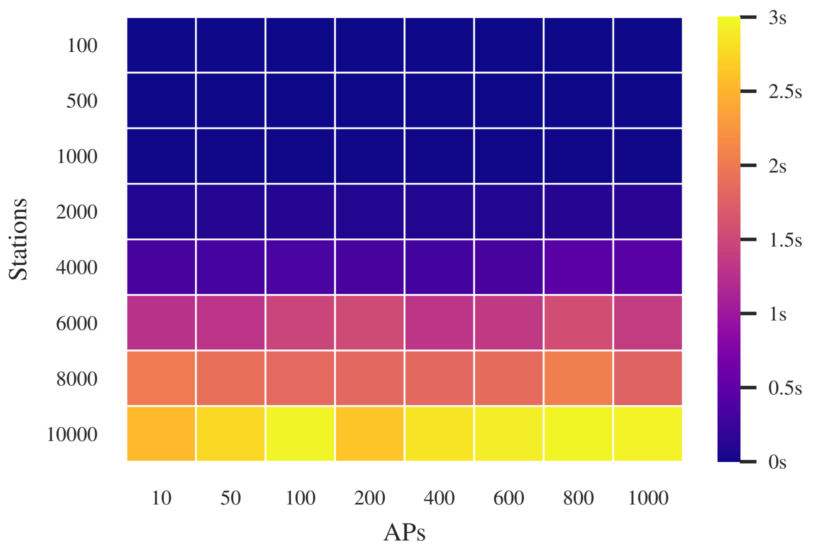


Figure 5.7: Scalability of the greedy heuristic in terms of stations and APs.

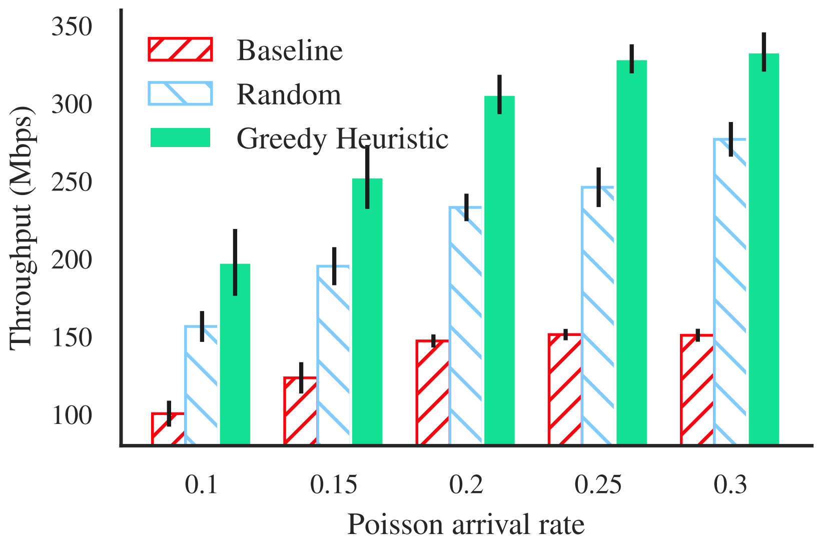


Figure 5.8: Throughput as a function of poisson parameters, error bars depict the standard error, for short flow lengths between 5 and 15 s.

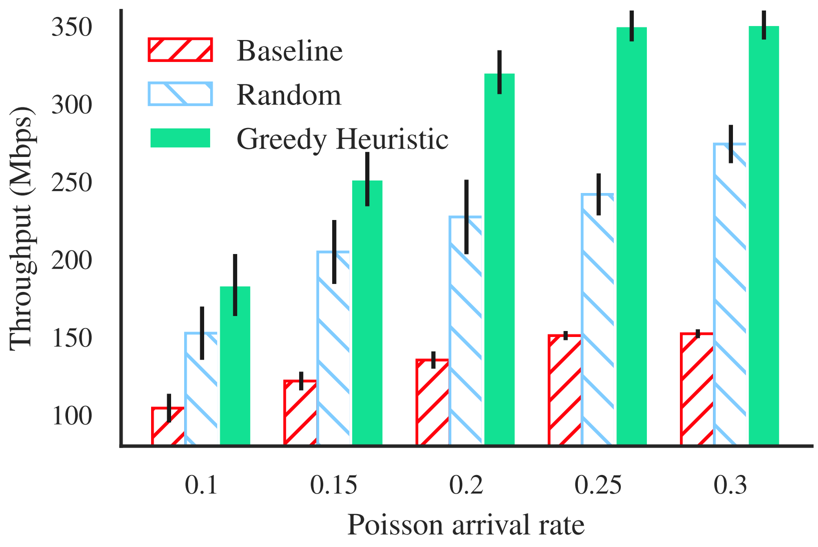


Figure 5.9: Throughput as a function of poisson parameters, error bars depict the standard error, for long flow length between 10 and 30 s.

Figures 5.8 and 5.9 show that for all different arrival rates, the greedy heuristic approach significantly outperforms the distributed baseline and the random approach, across both scenarios. For the first scenario we see that for 0.1 as Poisson interval, the baseline random algorithm, respectively, achieve a throughput of 100.60 Mbps (±9.23) and 156.65 Mbps (±9.91), while the greedy heuristic allows for a network-wide throughput of 197.87 Mbps (±21.48). When using a parameter value of 0.25, throughputs of 151.07 Mbps (±2.94), 246.18 Mbps (±12.73), and 328.77 Mbps (±9.35) are achieved, respectively for the baseline, random algorithm, and heuristic approach. Equivalently, for the second scenario we see that for 0.1 as parameter value, the baseline achieves a throughput of 104.43 Mbps (±9.23), the random algorithm attains 152.61 Mbps (±17.13), while with the heuristic 183.58 Mbps (±19.94) is achieved. For 0.25 as parameter value, the baseline and random approach achieve, respectively, a throughput of 152.30 Mbps (±2.86) and 241.87 Mbps (±13.51). In contrast, the algorithm allows for a throughput of 351.02 Mbps (±9.58). This is a gain of, respectively, 130.48 % and 45.13 %. Similarly to the experiments with varying network loads, the throughput of the baseline and heuristic does not further grow for the last parameter value (of 0.3), as the total network capacity has been reached. Finally, note that we have also repeated this experiment for other ranges of flow lengths, but the results were nearly identical and are therefore omitted.

5.5 Conclusion

We have addressed in this chapter the combined problem of load balancing mobile connected devices across different connection points and scheduling flows across different technologies. To this extent, we have presented three algorithms. First, an optimal MILP formulation was proposed. Afterwards, a near-optimal heuristic that sequentially solves two mathematical formulations and a greedy heuristic were presented. We compare the performance of the three approaches using NS-3 simulations in a variety of scenarios. We showed that the greedy heuristic has the best scalability, up to networks with 10000 devices, while an improvement of potentially more than 100 % is possible, towards the fully distributed baseline, depending on the scenario.

6 Recognition of traffic patterns in the wireless spectrum

"If the road is easy, you're likely going the wrong way."

–Terry Goodkind (The Sword of Truth Series: Soul of the Fire, 1999)

The contributions presented in this chapter are based on the publication titled *"Detection of traffic patterns in the radio spectrum for cognitive wireless network management"*.

6.1 Introduction

In the previous chapters, we have shown how different, mostly wireless, networks can be managed in a more dynamic and intelligent manner to increase user experience and network performance. This intelligence is based on real-time monitoring information, provided by underlying management solutions such as ORCHESTRA (cf. Chapter 3). However, we focused on manageable networks and did not yet look at coping with other coexisting, uncontrollable, networks or technologies. The later becomes grows more important as the wireless spectrum is becoming (over)crowded, especially in the unlicensed frequency bands, due to the continuous development and deployment of new communication technologies and increasing amounts of traffic. These different wireless technologies coexist next to each other at identical or overlapping physical locations, often competing for the same finite and limited radio spectrum resources such as bandwidth and air-time. This situation combined with the increasing traffic demands can often lead to significant

QoS degradations and performance loss [19]. This is, among others, the case for LTE-U and Wi-Fi in the 5 GHz frequency, as well as Wi-Fi, ZigBee, and Bluetooth in the 2.4 GHz band. In order to address this problem, it is clear that there is a need for intelligent and more efficient use of spectral resources [10, 36]. Furthermore, in order to provide the management solutions and optimization algorithms with the necessary information about the state of the spectrum, in order to cope with neighboring networks and technologies, fine-grained monitoring systems are required.

The principle of Cognitive Radio (CR) has been introduced to allow the coexistence of different technologies and networks in the same spectrum bands [36]. One of the most known methods to increase spectral efficiency, within the domain of CR, is Dynamic Spectrum Access (DSA) [36]. In order to perform DSA, the sensing capability of the CR system is fundamental as it allows to identify if an (un)known technology is accessing the spectrum at the same frequency and time. As such, appropriate measures can be taken to combat performance degradation due to interference or collisions. However, detecting a given technology is often not enough to increase spectrum efficiency, as no information is provided about the exact spectrum usage within the transmitted signal of the technology in question.

One way to solve this problem is by augmenting the identification of technologies with information about the network's traffic patterns. However, traffic identification is traditionally performed by using intrusive methods based on Deep Packet Inspection (DPI) and (IP) packet traces [116, 117]. These methods are typically available when the listening or monitoring device is part of the network. In contrast, we believe that a better approach is to learn the traffic behavior of the other networks directly by observing the spectrum and then offload own traffic during the interference-free parts of the spectrum, given the learned pattern. This allows to significantly increase spectrum efficiency, especially in places with a dense and overlapping presence of wireless networks. To this extent, we explore the option of performing traffic recognition directly using spectral data. To the best of our knowledge, this is a novel research direction that has not been studied before.

In particular, we present a Deep Learning (DL)-based approach to recognize different traffic patterns. A Convolutional Neural Network (CNN) architecture is designed that forms the base of three different prediction models that can differentiate TCP and UDP traffic, recognize constant and burst traffic with different duty cycles, and identify different transmission rates. This strongly contrasts the aforementioned more intrusive methods, as the listening device does not need to be part of the network and modern privacy requirements are respected. Furthermore, as mentioned in Section 2.4.1, existing ML-based approaches for traffic recognition typically operate on packet traces obtained from wired networks. Since existing CR approaches focus on the recognition of different technology or modulation schemes, we present the first approach to detect traffic information on spectral data. Additionally, we combine two state-of-the-art simulators, namely the NS-3 network simulator and the Matlab toolbox, in a domain randomization approach. This allows generating, relatively easy, large amounts of synthetic data that can be used to train more robust models, capable of coping with the uniqueness of different wireless environments. The performance of these synthetically trained models is evaluated in a real network environment.

The remainder of this chapter is structured as follows. First, we introduce the problem and construct the different traffic recognition models in Section 6.2. Next, we present our data collection framework in Section 6.3. Finally, we present an evaluation of the different traffic models in Section 6.4.

6.2 Traffic recognition models

In this section, we start with the introduction of a formal problem definition of recognizing traffic patterns at a spectral level. Afterwards, we discuss how the spectrum data is used and define the necessary labels. We round up this section by presenting the shared CNN architecture and the three different prediction models.

6.2.1 Problem definition

Our goal is to explore if it is possible to detect traffic patterns directly in the wireless spectrum. We propose to use a Supervised Learning (SL) approach to see if patterns can be detected at all. The following problem formulation is defined:

Let $X = \{x_1, x_2, \ldots, x_N\}$ and $Y = \{y_1, y_2, \ldots, y_N\}$ be the sets of N examples and their corresponding labels, respectively, where $x_i \in X$ and $y_i \in Y$ for all $i \in [N] := \{1, 2, \ldots, N\}$. In SL, the goal is to learn a mapping from X to Y given a training set of pairs (x_i, y_i), where y_i is the label of the ith example x_i. In other words, given the sets X and Y, SL tries to find a function f such that $y = f(x)$.

Within the scope of performing traffic recognition using spectrum data, the problem is to find a function f that given a representation of the spectrum $x_i \in X$ (e.g., raw IQ samples or Short Time Fourier Transform (STFT) of the IQ samples), is able to predict $y_i \in Y$. In this case, Y is a set of labels that represent the properties of the transmitted traffic that we are trying to identify. These traffic properties can, among others, be the employed transport protocol, transmission rate, or the traffic pattern of the application (e.g., burst or constant). In the next sections, we propose to use DL architectures to solve the above-described problem. In particular, we employ CNN models to build the function f through a combination of many less complex functions that are connected hierarchically [164].

6.2.2 Input spectrum representation and traffic labels

Before designing and implementing a CNN for traffic recognition tasks using spectrum data, we need to define the type of input data that represents the radio spectrum. A popular method within the area of CR is the use of IQ samples. IQ refers to two sinusoids in which a radio signal can be decomposed. Those two sinusoids have the same frequency and are 90° out of phase (hence *in quadrature*). By convention, the I signal is a cosine waveform, and the Q signal is a sine waveform. For completeness, the mathematical formulation for this can be seen in the following equation:

$$s(t) = i(t) + q(t) = I(t)\cos(2\pi f_c t) + Q(t)\sin(2\pi f_c t)$$

With $s(t)$ the original transmitted signal, $i(t)$ the in-phase component, $q(t)$ the quadrature component, and f_c the center frequency of the carrier. This approach is widely used due to its relatively simple mathematical operations and its flexibility to generate any modulation scheme by combining the appropriate I(t) and Q(t) values. Note that IQ signals are always modulated based on the amplitude.

Typical CR tasks that make use of IQ samples are technology and modulation identification, as highlighted in Section 2.4.2. These tasks discriminate between different classes (e.g., different technologies) using features that are visible in short periods (at most hundred of microseconds) [132,133]. For instance, let us consider the task of modulation recognition, as presented by O'Shea et al. [132]. As input, 128 IQ samples are used that represent 128 us of the signal at a sampling rate of 1 million samples/s. However, in contrast to these related tasks, traffic recognition is based on traffic features that are only visible in longer periods of time (at least in a magnitude of hundreds of milliseconds). If we would consider the same sampling rate as in the aforementioned example, representing 0.5 s of spectrum data for traffic recognition would require 0.5 million of IQ samples. However, implementing ML models that utilize this input size is impossible, due to the enormous amounts of memory, computational resources, and storage required in order to train and run such models.

With the need of having a broader view of the spectrum in time, while keeping the input data as small as possible, it is clear that directly using IQ samples is not an option. To this extent, we propose the following data transformation: given a time window w (in seconds) and raw spectrum data as input (i.e., IQ samples), the amount of data collected during the interval w is plotted and saved as an image (in compressed format). In this work, we use a window size $w = 0.5$ s (experimentally determined), while the generated images are RGB (normalized between 0 and 1) with a size ([height,width]) of [656,874] pixels. Each pixel represents an averaged value of the transmission power observed at that particular moment. Furthermore, images are saved in the Portable Network Graphics (PNG) format. As a result of this data transformation, the size of the images, generated during this research, varied in a range between 3.3 KB and 114 KB. In general terms, the required storage for the image dataset is less than 1 % of the dataset based on raw IQ samples. In Section 6.4, we provide more details on the resulting datasets. Intuitively, the idea of converting IQ samples to images as input for DL models make sense, due to the high success of applying DL techniques on pixel inputs [165–167].

Finally, in order to generate labels for the input dataset, we focus on the recognition of three different properties of the generated traffic: the transport protocol (TCP versus UDP), the traffic pattern (constant versus 3 types of burst traffic with a duty cycle of, respectively, 25, 50, or 75 %), and the rate at which the traffic was transmitted (100 Kbps, 1 Mbps, 10 Mbps, or 50 Mbps). Each generated image is associated with a label for each of the three properties. More details on the process and framework to generate the data are presented in Section 6.3.

6.2.3 Convolutional Neural Network for traffic recognition

In order to exploit the new representation of the input data as images, we design and implement six different DL models based on a CNN architecture. CNN architectures are specifically designed for, and have proven their worth in, many computer vision applications [168]. Moreover, they have become the de facto standard in the area of computer vision [169]. Under this assumption, the forward function (i.e., moving the data from the input to the output side through the neural network) is more efficient and its computational complexity for learning is lower than traditional Deep Neural Networks (DNNs) based on fully-connected layers. This is because a CNN connects the neurons of a given layer, called a convolutional (or conv) layer, with only a few neurons of the next layer, reducing the number of parameters that need to be trained. Furthermore, these kinds of DNNs have been shown, as indicated in Section 2.4.2, to perform well in other related tasks such as modulation recognition. These models work either directly with raw spectrum data as input (e.g., IQ or Fast Fourier transform (FFT) samples) or with image representations of that raw spectrum data [130, 132].

We design a baseline architecture that is shared by all the models. In total, there are 6 models. For each classification task (three in total) there is a model for both time domain data and time-frequency domain data. Overall, the architecture consists of 5 different sequential blocks of layers, as can be seen in Figure 6.1. The first three blocks, each consisting of 5 different layers, are used to detect features. These blocks are built up by the following components: first, a convolution layer that contains a number of feature maps (also known as kernels or filters). These kernels are weighted matrices that are trained to detect and highlight certain features (i.e., object shapes) in the data. The number of kernels increases throughout the convolution layers of the three blocks, in order to step-by-step learn more detailed features. Next, is a Rectified Linear Unit (ReLU) activation function, followed by batch normalization and a Maxpooling function. ReLU is these days the most common activation function, as it converges fast, is cheap to compute (in terms of computational resources), and does not suffer from the vanishing gradient problem in deep neural networks [170, 171]. The pooling layer downsamples the given input, in order to further reduce the number of parameters, while also selecting the most relevant (hence the Maxpooling) inputs for the next block [172]. The fifth dropout layer is employed to increase generalization, as it randomly disables some of the neurons depending on the dropout rate (as shown in Table 6.2). This forces each of the remaining neurons to learn different features, increasing overall robustness. Each of the three blocks of layers is capable of detecting features at a certain level. Stacking multiple of such blocks (i.e., convolutional layers) allows for more complex features to be detected [172]. Note that, by stacking different layers, the chance of overfitting increases, as well as the amount of computational power and time required to train the models. The chance of overfitting

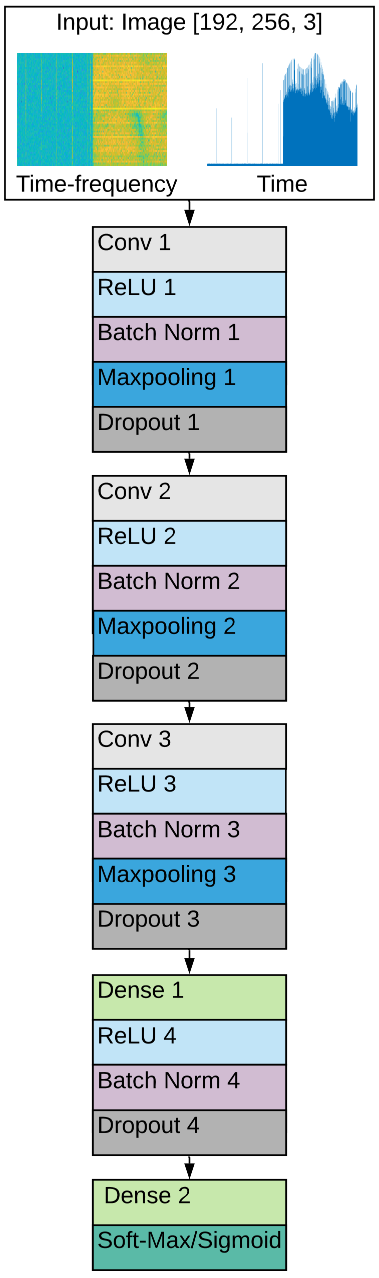


Figure 6.1: CNN architecture for traffic recognition.

can be reduced by using large amounts of data or generalization techniques such as the aforementioned random dropout. Furthermore, the last two blocks are used for classification. The fourth block is a dense layer with, once again, ReLU as the activation function. This layer also transforms the output of the last block of convolutional layers into a one-dimensional vector. Afterwards, this output is delivered to the last block, which is composed of a dense layer where the number of neurons equals the number of predefined labels for classification. If it is a binary classification (e.g., to distinguish between TCP and UDP), a sigmoid activation function is employed to map the input to the predefined labels. Otherwise, for the case of multi-class classification (e.g., for the transmission rate and duty cycle recognition models), a soft-max layer is used.

Figure 6.1 shows an overview of the resulting architecture, while Table 6.1 and Table 6.2 list the parameters used on each layer per model. Some specific parameters for the optimizer (e.g., the optimizer, learning rate, and batch size), for each layer (e.g., the number of filters, pool size, and dropout rate), and optimizer training, were determined using hyperparameter swapping. In total, the constructed CNN models have 25.2 million trainable parameters. We trained the models during 100 epochs using the Adam optimizer with a learning rate of 0.0005, cross-entropy loss function, and mini-batches of 128 RGB images [173]. These images were resized to [height,width] = [192,256] pixels due to memory constraints in the Graphics Processing Unit (GPU). We implemented our models using the Keras library and TensorFlow as the back-end [174, 175].

6.3 Data collection framework

One of the key challenges faced for the construction of any ML model, is the collection of sufficient amounts of quality data, needed for training and validation. In Section 2.4 we have mentioned how traditional traffic recognition approaches are typically operating or being trained, with wired packet data. Furthermore, as existing CR approaches are focusing on technology and modulation recognition, no accurately labeled datasets for traffic recognition in the wireless spectrum are currently available. To this extent, we present a new data collection framework in this section. We start by giving a motivation for our approach and, afterwards, present the architecture and features of the framework.

6.3.1 Motivation

A key concern is that the characteristics of wireless networks (e.g., interference, latency, or topology) tend to differ significantly across different environments. This means that it is not straightforward to train a model in a general fashion, applicable to a wireless context. Typically, this requires large volumes of training and validation data, gathered at different physical locations. Acquiring these amounts of data is cumbersome and time-consuming. To this extent, the use of simulation environments is very appealing as it allows to collect potentially large amounts of

Model	**Data Domain**	**Classification Task**	**# Kernels**	**Kernel Size (all Conv layers)**	**Pool size (all MaxPool layers)**	**Dense 1 # Neurons**	**Dense 2 # Neurons**
CNN 1	Time	Traffic Pattern	Conv 1 : 16 Conv 2 : 32 Conv 3 : 64	5 x 5	2x2	512	4
CNN 2	Time-Fequency						
CNN 3	Time	Transport Protocol					2
CNN 4	Time-Fequency						
CNN 5	Time	Transmission Rate					4
CNN 6	Time-Fequency						

Table 6.1: Shared hyperparameters among all the models.

Dropout Rate	**CNN 1**	**CNN 2**	**CNN 3**	**CNN 4**	**CNN 5**	**CNN 6**
Layer 1	0.1	0.6	0.7	0.6	0.7	0.5
Layer 2	0.3	0.5	0.5	0.3	0.4	0.4
Layer 3	0.6	0.5	0.5	0.3	0.4	0.4
Layer 4	0.6	0.3	0.3	0.1	0.2	0.2

Table 6.2: Specific dropout rate per model.

(training) data in an easy, safe, and reliable manner without the need to perform physical experiments. As such, limiting the effort of data collection. However, the behavior of agents and models trained in simulator environments are often specific to the characteristics of the simulator [176, 177]. This is due to the errors and abstractions made by modeling the complex real world.

Recently, it has been proved that this so-called reality gap, between the real-world and the simulators, can be bridged using a persevering form of randomization. This technique, called domain randomization, trains learning models based only on (low-fidelity) simulated data by randomizing all non-essential aspects of the simulator [176, 177]. One of the core ideas behind this approach is that by training on a wide enough variety of unrealistic procedurally generated samples, the learned models will generalize to realistic scenarios [177]. This main principle is illustrated in Figure 6.2 In other words, the model should consider reality, as just another adaptation as the model only captures the bare essentials [178]. As such, more robust models have been constructed that cope with more dynamic behavior in the real world [176].

Up to now, the technique has mostly been used in the area of robotics and self-driving vehicles where typically cameras and computer vision techniques are being used for object recognition. One of the main applications so-far has been robot grasping, where a robot needs to be trained to pick-up or place certain objects. Tobin et al. have shown that their grasping model, trained entirely using non-realistic generated objects in simulation, can achieve high success rates: a successful grasp was observed for 96 % upon the first 20 samples, while in general a success rate of 92 % was achieved on realistic objects on the first attempt [177]. Furthermore, applications and models for object recognition (e.g., detection of other cars in an autonomous vehicle use case) have shown similar success rates. Here, aspects such as lightning, pose, object textures, and colors are randomized in non-realistic ways to force the neural network to learn the essential features [178, 179]. A key observation is that these models could be used in real-world environments without any retraining.

It is clear that domain randomization is an emerging and promising research area. As such, we will further explore this principle in the significant different

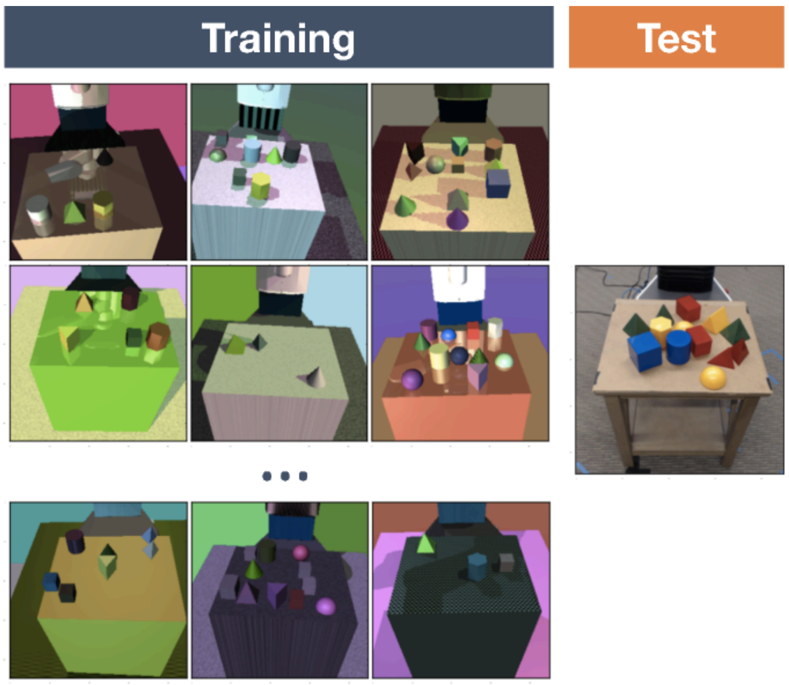


Figure 6.2: Illustration of the domain randomization approach [177].

application domain of wireless networks. We want to investigate if by using this approach, a robust traffic recognition model can be trained that can cope with the dynamic wireless context. In the next section, we will present a data generation architecture, as the first step towards a full data randomization approach.

6.3.2 Architecture

In Section 6.2.2 we have discussed that images, generated based on captured IQ samples, are used as input data for the different models. This means that a simulation environment is needed that can generate such IQ samples at the physical layer of the OSI stack. Furthermore, since we want to recognize traffic patterns, the simulator should also contain the logic of the higher layers (e.g., transport layer) of the OSI stack. Unfortunately, no simulator currently exists that extensively covers all different layers of the network stack. As such, we combine two state-of-the-art solutions that each cover a part of the network stack. We use the discrete-event network simulator NS-3 (version 3.29) to cover the traffic generation and the higher layers of the stack [160]. This includes the transport layer that, for instance, contains TCP rate control mechanisms. Regarding the lower layers, we consider the Matlab toolbox, in particular, to address the physical layer, as the toolbox can generate wireless signals and Radio Frequency (RF) spectrograms

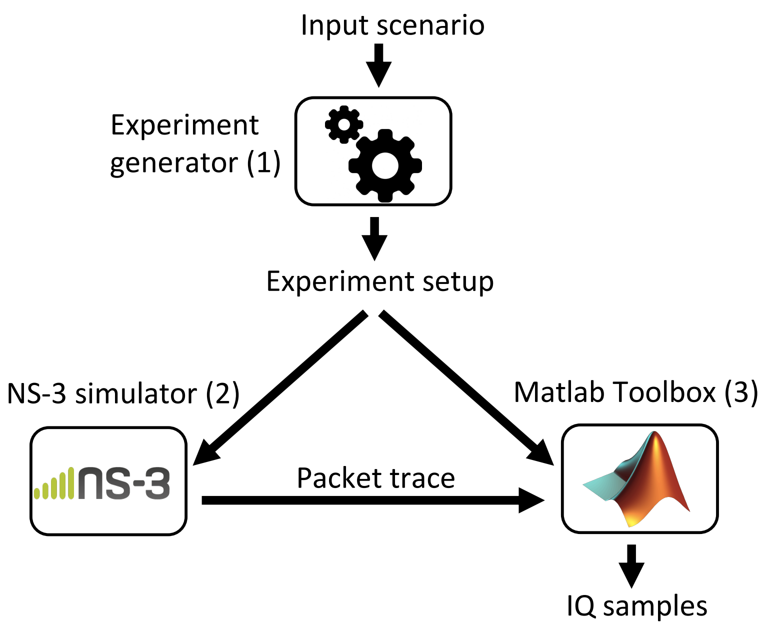


Figure 6.3: Architectural overview of the data generation framework.

(i.e., IQ samples) [180]. We make use of the WLAN toolbox version 2.0 integrated with Matlab R2018b.

The overall architecture of the proposed framework is depicted in Figure 6.3. As can be seen, there are three main components. First, there is the experiment generator that receives as input a scenario description in JSON format. This scenario description contains for several selected parameters a list of values to be considered in the simulations. The tunable parameters are depicted in Table 6.3. Note that this formulation is easily extendable to include other features and parameters (e.g., MIMO, packet aggregation, or more complex applications). The experiment generator (1) generates an experiment setup for each unique combination of all parameter values. For instance, consider the (theoretical) scenario where two parameters A and B are specified, with the parameters having, respectively, the following values: (A1, A2) and (B1, B2, B3). The experiment generator will in total create six unique configurations (A1B1, A1B2, A1B3, A2B1, A2B2, A2B3). Note that this experiment generator can potentially create a very large amount of combinations, under a very large parameter space, making it infeasible to verify all unique combinations. While this is not the case for the work currently presented in this chapter, future work could explore surrogate modeling techniques to cope with these massive data spaces.

Next, based on the generated experiment setup, the NS-3 simulator constructs the desired network topology with the specified technology standard and the number of stations and APs. During the length of the simulation, the specified traffic

Considered parameters	
Simulation	simulation length
Network topology	number of the nodes, their locations, and mobility patterns
Traffic descriptions	number of flows, transport protocol, traffic patterns, and rates
Technology (Wi-Fi parameters)	MCS, channel width, standard, and guard interval

Table 6.3: Overview of tunable parameters for data generation.

is generated through this network, while stations can move (if applicable) based on the selected mobility pattern. Furthermore, a log file is generated that contains all the transmitted packets and transmission information needed to generate the transmitted waveform in Matlab. This log can be seen as a regular packet dump (cf. a pcap file), but augmented with the following information: the specific type of packet (e.g., beacon, association request, data), technology specific information (e.g., for Wi-Fi: STBC, PSDU length, AMPDU), and the MCS value.

Afterwards, Matlab (3) performs the following three steps: first, a MAC packet is generated per line of the previously generated trace file in combination with the information provided. This MAC packet is used to generate a waveform that is compliant with the 802.11 standards. The waveform is expressed as an array of complex numbers that represent the IQ samples of the signal. After each packet is generated, the IQ samples are stored in binary format in a file. Note that the IQ samples are, at this stage, free of noise and do not include any channel effects. Second, the generated IQ samples are augmented by passing them through both a modeled fading channel (according to the 802.11 standards) and an Additive White Gaussian Noise (AWGN) channel. The latter adds white Gaussian noise to the signal, with an SNR from 0 dB to 30 dB in steps of +3 dB. This means that per original IQ sample, an additional amount of 12 IQ samples are created: 1 with only channel fading effects, and 11 with channel fading and noise (each with a different SNR). This is done in order to randomize the spectrum conditions. As such, following one of the main principles behind the domain randomization technique, as discussed previously. Third, based on the previously created IQ samples, we plot them in time (amplitude of the IQ samples) and time-frequency (by applying the STFT on the IQ samples) domain and save them to disk. This leads to two different input datasets, which are compared in Section 6.4. Note that each created image represents a traffic snapshot of size w = 0.5 s, which is equivalent to 0.5 million of IQ samples. To clarify, imagine an experiment length of 5 s. Since we take snapshots of 0.5 s, 10 (experiment length divided by snapshot size) $\times$ 13 (the clean image + the additional ones with channel fading and noise effects) images are created for the specific scenario, for both time and time-frequency domain. Additionally, we want to highlight that the presented approach is only a first step towards a full domain randomization approach, as a complete domain randomization approach also considers directly changing the classification models and the use of non-realistic randomized data.

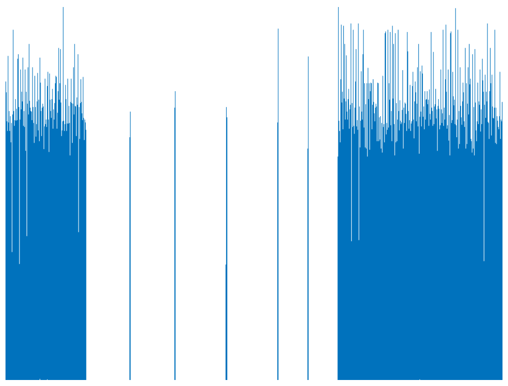
To illustrate the generated datasets, we depict for both the time and time-domain dataset three images in, respectively, Figures 6.4 and 6.5. The images are one of the snapshots made for a scenario with a UDP traffic flow of 10 Mbps and a duty cycle of 75 %. Each figure is composed of three images that respectively the different steps in the data adaptation process, as described above. The top image shows the created snapshot of the original signal based on pure IQ samples). Furthermore, the middle image shows the effect of adding 802.11 channel effects to the signal, while the adding of noise to the signal, using an SNR of 15 dB.

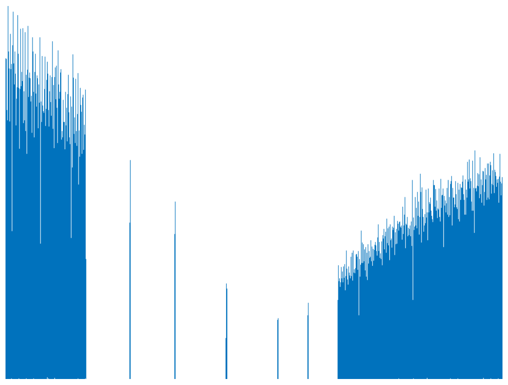
6.4 Evaluation and discussion

In this section, we evaluate the accuracy of the three traffic recognition models, while also investigating the use of the models in real-life settings. First, we discuss the details of the generated datasets for the training and validation of the presented models. Next, we validate the models using data collected in a real-life setting. Afterwards, we demonstrate the real-life usage of the models through a small-scale prototype.

6.4.1 Description of generated training datasets

In order to evaluate the designed models, we use the data collection framework presented in Section 6.3.2 to generate the training, validation, and test datasets used for all experiments presented in Section 6.4.2. As a topology, we assume one station connected to a Wi-Fi AP at a distance of 5 m. A single flow of traffic was transmitted for 5 s between the two devices with varying rate (100 Kbps, 1 Mbps, 10 Mbps, and 50 Mbps), transport protocol (TCP, or UDP), and transmission pattern of a so-called On-Off application (with 25, 50, 75, 100 % duty cycle). Furthermore, we varied the following Wi-Fi parameters: the standard and frequency (802.11n on 2.4 Ghz, 802.11n on 5 Ghz, 802.11ac on 5 Ghz), the channel width (20 Mhz,40 Mhz), and guard interval (short, long), while assuming the presence of the dynamic Minstrel Rate control algorithm (i.e., dynamic MCS values). Based on these parameters 384 unique experiments were constructed and augmented, leading to over 800 GB of IQ samples. As discussed in the previous section, these samples are processed into two image datasets, time and time-frequency domain, each one composed of 49920 images and requiring around 6 GB of storage. This transformation reduces the storage requirement to less than 1% of the required storage of the raw IQ samples dataset. Finally, each dataset was randomly split to create the training (80%), the validation (10%), and the test (10%) datasets. In the next section, we compare the performance of all constructed models, for the three classification tasks using both datasets.


(a) Original signal (no channel or noise effects)


(b) Signal with channel effects

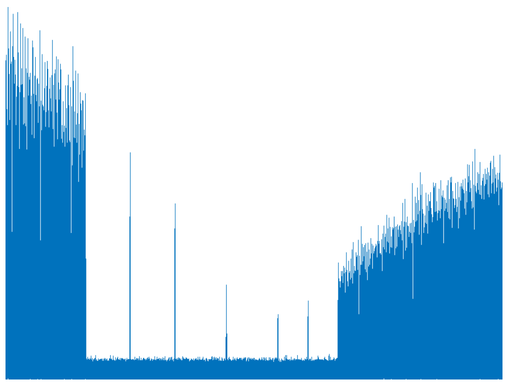
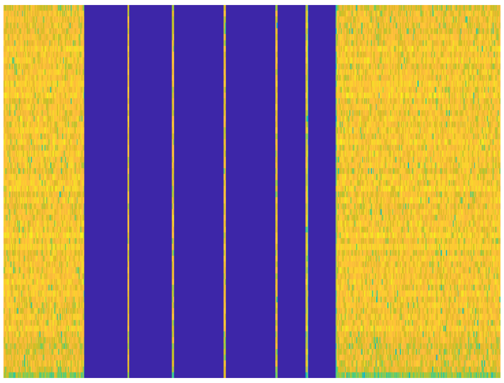
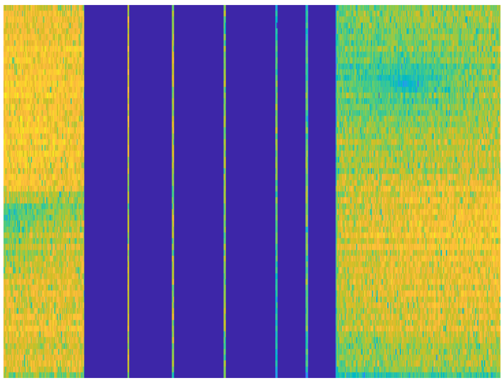

(c) Signal with channel and noise (SNR of 15 dB) effects

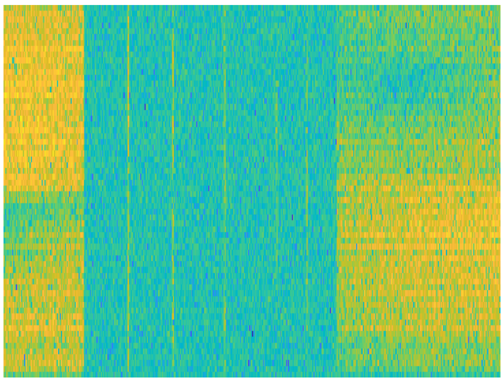
Figure 6.4: Three images in time domain with different channel and noise effects.



(a) Original signal (no channel or noise effects)



(b) Signal with channel effects



(c) Signal with channel and noise (SNR of 15 dB) effects

Figure 6.5: Three images in time-frequency domain with different channel and noise effects.

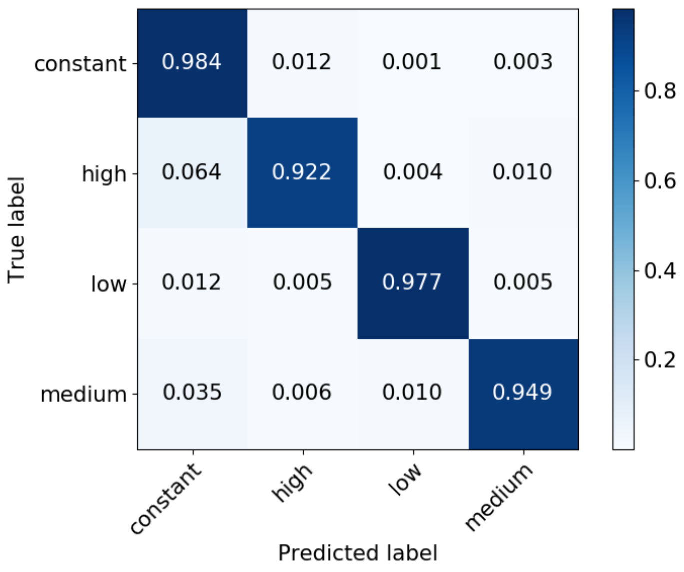
Model			Accuracy		
			Train	**Validation**	**Test**
CNN 1	Traffic	Time	0.999	0.962	0.962
CNN 2	Pattern	Time-Frequency	0.999	0.991	0.990
CNN 3	Transport	Time	0.995	0.965	0.968
CNN 4	Protocol	Time-Frequency	0.999	0.998	0.997
CNN 5	Tx rate	Time	0.987	0.979	0.978
CNN 6		Time-Frequency	0.999	0.999	0.998

Table 6.4: Accuracy of all three models, using training, validation, and test datasets.

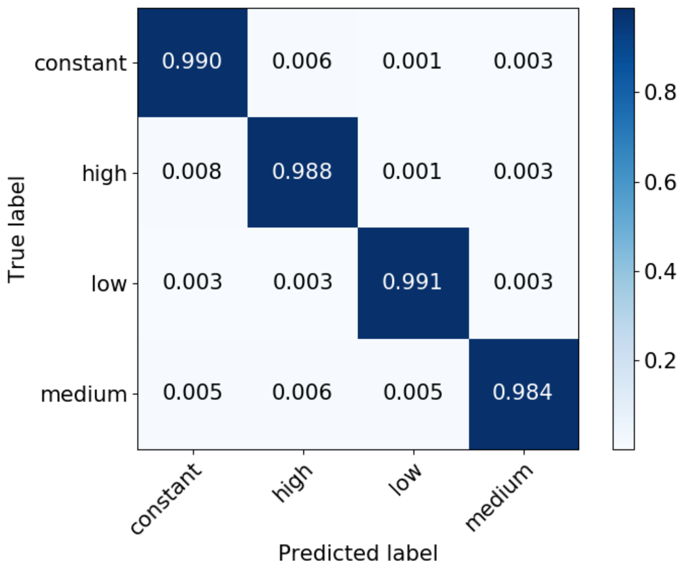
6.4.2 Evaluation using generated synthetic data

Table 6.4 depicts the accuracy for each model, obtained with respectively, training, validation, and test datasets, for both time (amplitude) and time-frequency (STFT) domains. Overall, it is clear that all models, under all datasets, have an accuracy of above 96 %. Below we discuss the results for each model individually and show the resulting confusion matrices. These matrices, also known as error matrices, show for each model how all data samples are classified. As such, additional insights into the performance of the models are acquired.

As mentioned in Section 6.2.2, two models are trained to recognize 4 different traffic patterns (burst versus constant) with a duty cycle of 25 % (low), 50 % (medium), 75 % (high), or 100 % (constant). Table 6.4 shows the final accuracy of the training, validation, and test datasets after training. CNN 1 denotes the model trained with the dataset in time domain, while CNN 2 is the model that is trained with time-frequency data. Both models, time and time-frequency, achieved above 96 % accuracy in validation and test. However, using time-frequency data leads to higher accuracies above 99 %. This difference, confirmed by other literature as well, is due to the fact that (high amounts of) noise together with fading channel effects have a bigger impact on images that represent the time domain [181]. Note that creating images in time domain is faster, as it requires less complex mathematical computations than STFT. Figure 6.6 shows the confusion matrices, respectively, for time and time-frequency domain data, obtained from evaluation with the test dataset. According to the confusion matrices shown, the medium and high traffic patterns were the hardest to discriminate, and some of the examples were miss-classified as constant traffic. This is the case for both the time and time-frequency domain. This behavior can be explained due to the MCS adaptation algorithm. If the transmission of a signal is using a lower MCS, then more spectrum will be used in comparison to the same signal using a higher MCS. This in respect to a certain transmission rate and the time window w that is used to create



(a) Using time domain images



(b) Using time-frequency domain images

Figure 6.6: Normalized confusion matrices for the traffic pattern model.

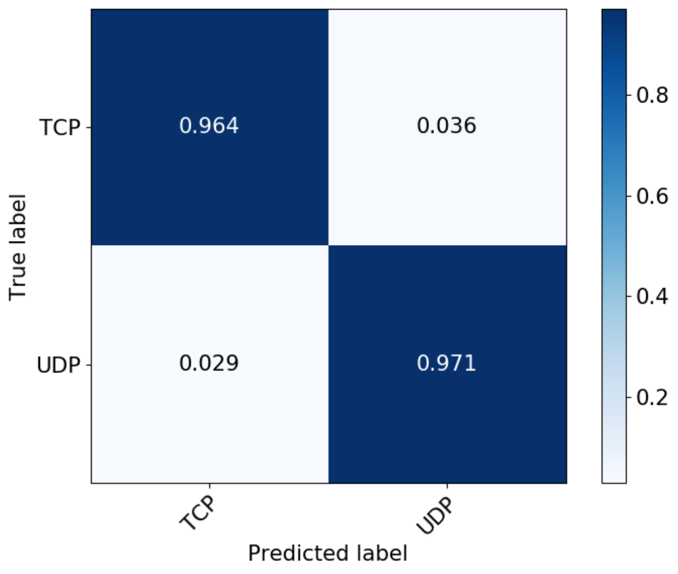
the images, in our case of $w = 0.5\,\text{s}$.

For the task of discriminating between TCP and UDP traffic, we created two models, one for time domain images and one for time-frequency domain data, to discriminate between two different transport protocols (UDP and TCP). In Table 6.4 we can see how both models have an accuracy more than 96 % in validation and test. An accuracy of more than 99 % is even possible when using time-frequency images. This is similar to the trend noticed above for recognition of different traffic patterns. Figure 6.7 shows the confusion matrices, respectively, for time and time-frequency domain data, obtained from evaluation with the test dataset. These confusion matrices indicate that the miss-classification distribution among the two classes is very similar. This result can be explained by the fact that for scenarios with large traffic flows (i.e., high transmission rates and duty cycles), the spectrum is very occupied and the snapshots and resulting images tend to be more similar for both the TCP and UDP protocols.

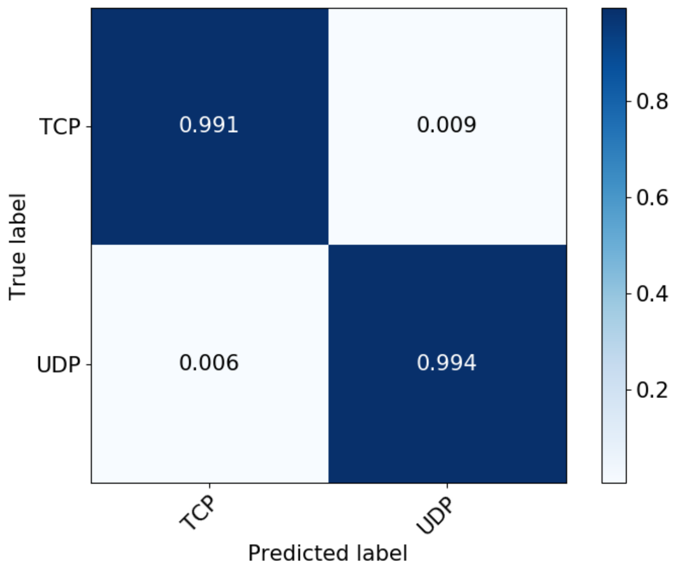
For the third classification task of recognizing different traffic rates, we also designed two models that can recognize the transmission rate of a transmitted traffic flow, based on either time or time-frequency image datasets. The results in Table 6.4 and Figure 6.8 show that it is possible to accurately discriminate between different TX rates. In Table 6.4, CNN 5 denotes the model trained with the dataset in time domain, while CNN 6 is the model that is trained with time-frequency data. CNN 5 has an accuracy for the test dataset of 97.8 %, while CNN 6 achieves a classification result of 99.8 %. Furthermore, the confusion matrices in Figure 6.8 show that most of the miss-classification, across both domains, occurred for transmissions of 10 Mbps, which were miss-classified as signals generated at 50 Mbps. When using time domain images, rates of 100 Kbps were also sometimes miss-classified as 1 Mbps or 10 Mbps. Similar to the results of the aforementioned classification tasks, combinations of a high transmission rate with a high duty cycle can lead to samples that look very similar in the spectrum. This is especially true if they are augmented with fading effects and noise, which may be difficult to discriminate by the classifier. However, note these kinds of examples are also responsible for providing a better generalization capacity to the DL models in order to learn and generalize better to unseen data. This is also verified given the high accuracy in both validation and test datasets. Finally, note that the task of classifying traffic patterns and traffic rates can be replaced for more complex regression models in order to predict continuous values of these datasets, instead of using classification with predefined labels.

6.4.3 Validation using real-life data

In the previous section, we have used synthetic data, generated through our data generation framework, as test datasets to evaluate the different constructed models. As a next step, we perform an evaluation with data collected in a real-life setting. Besides giving additional insights into the performance of the models, this also presents us with an indication of the applicability of domain randomization within the context of wireless networks. We create a setup consisting of 4 devices: two

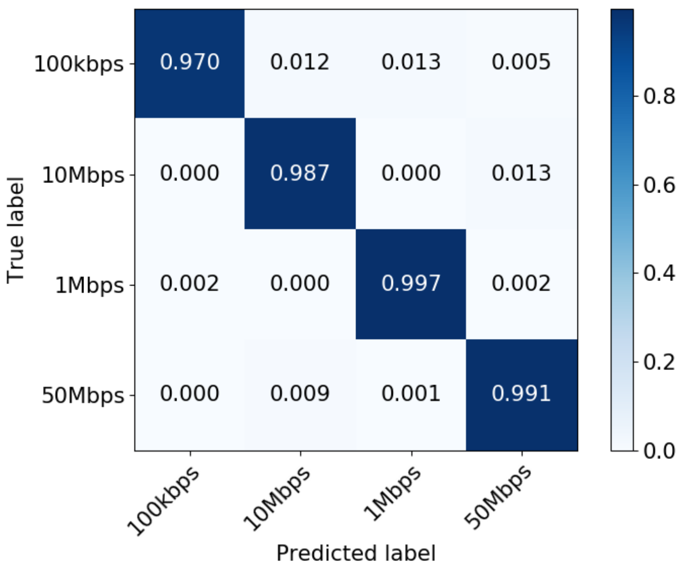


(a) Using time domain images

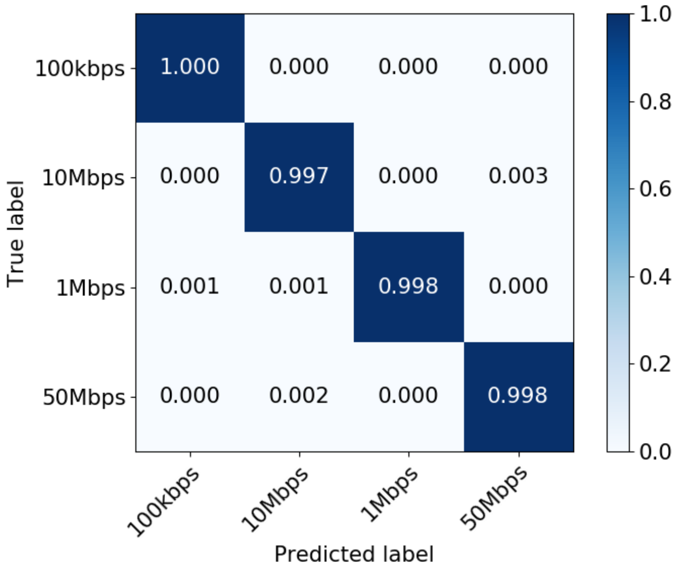


(b) Using time-frequency domain images

Figure 6.7: Normalized confusion matrices for the traffic protocol model.



(a) Using time domain images



(b) Using time-frequency domain images

Figure 6.8: Normalized confusion matrices for the traffic rate model.

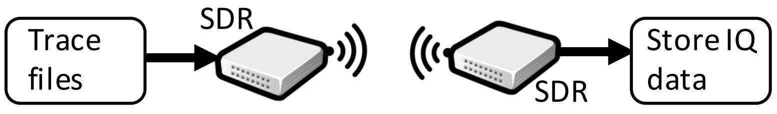


Figure 6.9: Overview of the real-life setup.

Intel NUCs and two SDRs. Each NUC has an i5 core processor, 16 GB of RAM, and an SSD of 500 GB. Each SDR is connected to one of the Intel NUCs, as can be seen in Figure 6.9. The NUC at the left-hand side is responsible for generating traffic that is transmitted by its connected SDR. The other SDR, at the right side of the figure, captures IQ samples that are stored on the second NUC. There is a distance of 3-4 m between the two SDRs. The captured IQ samples, are afterwards converted into time-domain (amplitude) and time-frequency (STFT) images, using a window size of 5 s. Note that this is the same procedure as for the synthetically generated datasets.

For generating the traffic, we make use of the IQ samples that were generated by our data generation framework. We replay those IQ samples using GNU Radio, installed on the NUC connected to the transmitting SDR, that is capable of, among others, transmitting IQ samples [182]. By replaying these samples, we can investigate the robustness of the different modules in a real-life setting, especially the impact of real channel fading and noise effects on the performance. Note that we use the pure IQ samples for this, before the addition of noise and channel fading effects. In particular, we make use of 32 traces that combine all values of the label of our classification models: 4 options for transmission pattern (25 %, 50 %, 75 %, or 100 %)), 2 options for transport protocol (TCP, or UDP), and 4 options for transmission rate (100 Kbps, 1 Mbps, 10 Mbps, or 50 Mbps). Finally, this experiment is conducted in a real-life channel but we checked with a spectrum analyzer for the absence of strong nearby interference sources.

Table 6.5 depicts the accuracy for each model, obtained with the real-life dataset, for both time (amplitude) and time-frequency (STFT) domains. Overall, it is clear that there is a significant difference in comparison to the accuracies reported in Table 6.4. Furthermore, the difference between the accuracies achieved with time domain and time-frequency domain has also grown. We discuss below the results per classification task.

The accuracies for the classification of traffic patterns are shown in Table 6.5 and the accompanying confusion matrices are presented in Figure 6.10. The model trained with the dataset in time domain achieves an overall accuracy of 54.1 %, while the model trained with time-frequency images obtains an accuracy of 59.1 %. These values are significantly lower than the ones obtained previously: respectively, 96.2 % and 99.0 %. The better accuracy is obtained by using the time-frequency dataset, as motivated previously. From Figure 6.10, we see that the model mostly miss-classifies the medium, high, and constant traffic patterns. More precisely, medium traffic is roughly in 1 out of 3 cases miss-classified as low traf-

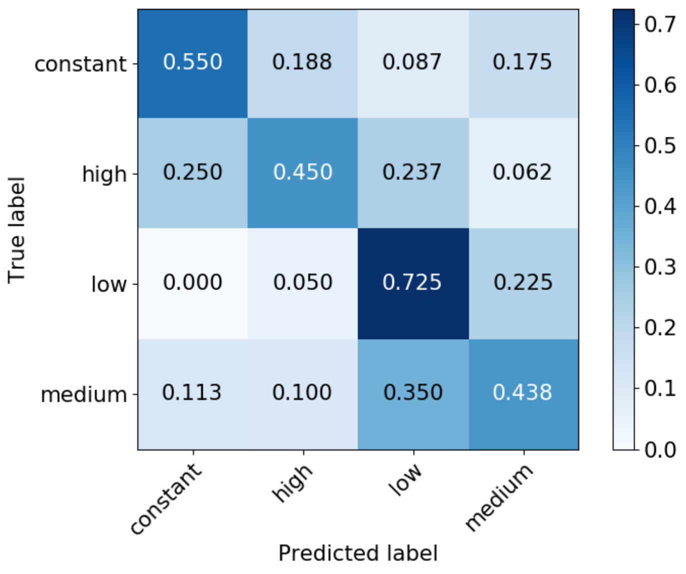
Model			Accuracy
CNN 1	Traffic	Time	0.541
CNN 2	Pattern	Time-Frequency	0.591
CNN 3	Transport	Time	0.591
CNN 4	Protocol	Time-Frequency	0.782
CNN 5	Tx rate	Time	0.734
CNN 6		Time-Frequency	0.869

Table 6.5: Accuracy of all three models using the real-life dataset.

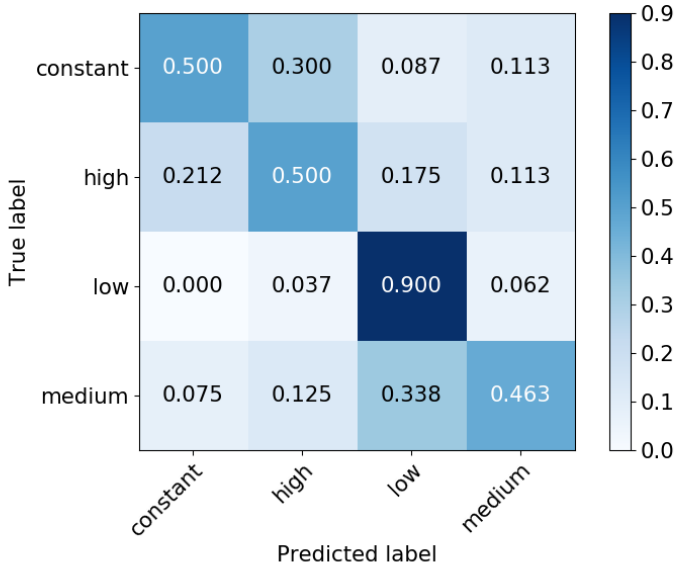
fic, while constant traffic is often miss-classified as high traffic and vice versa. This is the case for both models. Furthermore, note that these miss-classifications already occurred (at much lower scale) with the synthetic test dataset previously. We motivated that behavior by the impact of the MCS adaptation algorithm. Here, we can say that when using real-life data, the errors of the model are amplified.

Correctly discriminating TCP and UDP is successfully done in 59.1 % of the time for the model in time domain, and 78.2 % of the time for the time-frequency model. While both accuracies are still lower than reported previously with the test dataset (respectively, 96.8 % and 99.7 %), the difference between the two models is very large. The difference in accuracy for CNN 4 (time-frequency) using test or real-life data is roughly only half the size of the difference recorded for CNN 3 (time). When considering the confusion matrices in Figure 6.11, the difference between the two models is even more remarkable. The time domain model can better classify UDP traffic, with an accuracy of 72.5 %, while the time-frequency model can classify TCP traffic with an accuracy of 96.9 %. The latter value is relatively close to the accuracy obtained with the test data (99.1 %), as reported in Figure 6.7b. We believe that the reason for these results is two-fold. First, as mentioned previously, the spectrum looks rather similar for different scenarios with high transmission rates and duty cycles, making it harder to distinguish between TCP and UDP protocols. Second, the time-domain models and images are more susceptible to channel effects and noise. This could make TCP ACK transmissions harder to detect, as such explaining the significant lower accuracy of CNN 3 for TCP classification.

The third classification task of detecting different transmissions rates achieves clearly the best overall accuracies. As reported in Table 6.5, we see that CNN 5 (time) and CNN 6 (time-frequency) achieve, respectively, 73.4 % and 86.9 %. This in contrast to the values reported earlier for the test dataset: 97.8 % and 99.8 %, respectively. From the confusion matrices, shown in Figure 6.12, we can conclude that the number of miss-classifications increases when increasing the transmission rate. Across both models, the labels of 10 Mbps and 50 Mbps are the hardest to

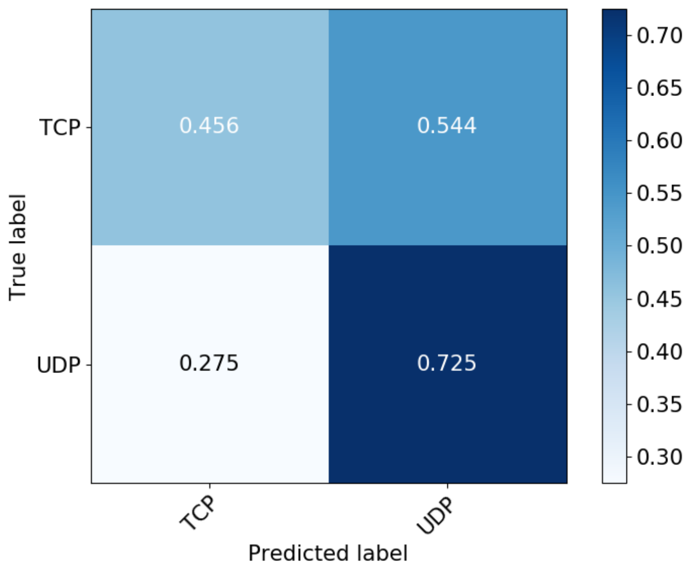


(a) Using time domain images

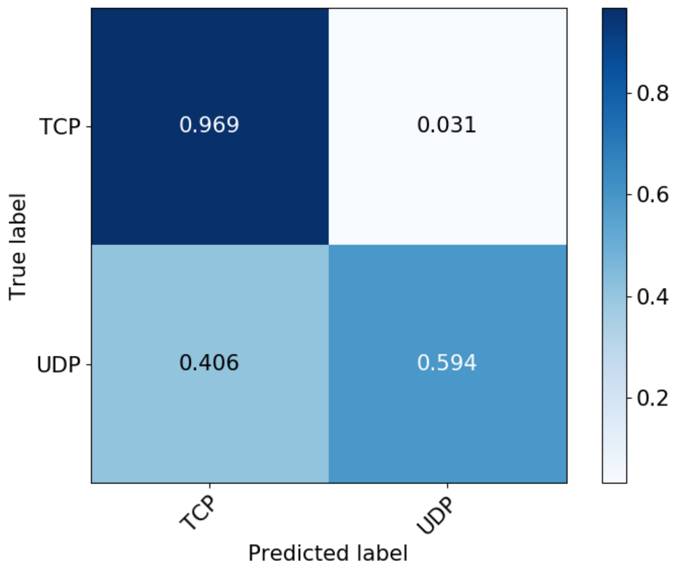


(b) Using time-frequency domain images

Figure 6.10: Normalized confusion matrices for the traffic pattern model.

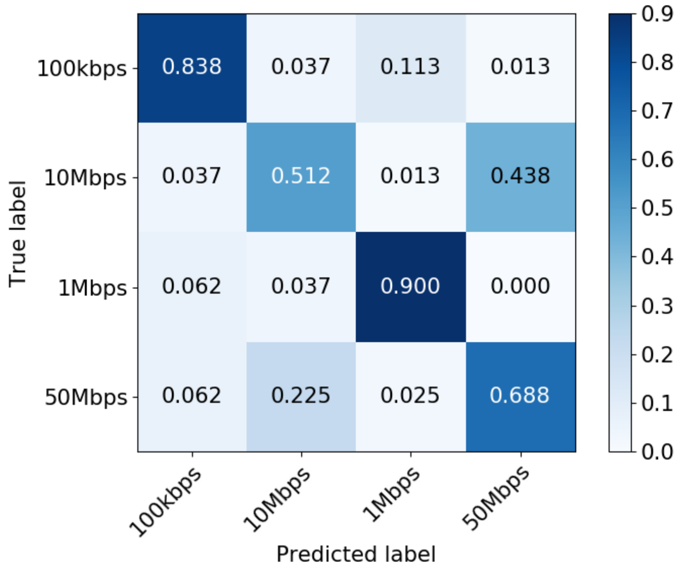


(a) Using time domain images

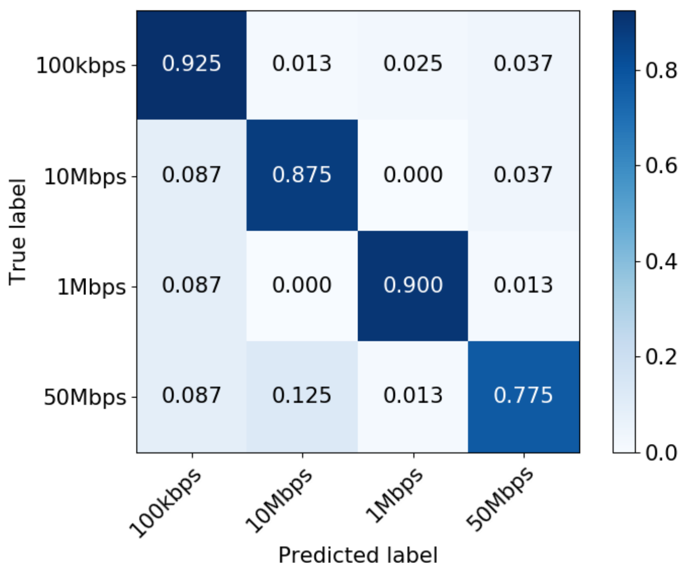


(b) Using time-frequency domain images

Figure 6.11: Normalized confusion matrices for the traffic protocol model.



(a) Using time domain images



(b) Using time-frequency domain images

Figure 6.12: Normalized confusion matrices for the traffic rate model.

classify. This can be explained by the strong similarity in the spectrum between both cases, as mentioned above.

Overall, we can conclude that the different models can cope relatively well with the unseen real-life data samples. This is especially the case for the classification of transport protocol and transmission rates, using time-frequency data (i.e., CNN 4 and CNN 6). However, we also acknowledge that the presented models should be further improved to boost overall accuracy. The latter is especially true for the classification of traffic patterns based on the duty cycle. When comparing the confusion matrices obtained from both synthetic and real-life validation, we notice that the already present miss-classifications of certain labels are enlarged upon using real-life data. We can conclude that the ideas behind domain randomization can indeed be applied to the recognition of patterns within wireless contexts and can help in making the models more robust. Future work should consider enhancing the models, in order to reduce the miss-classifications for both the synthetic and real-life datasets. Furthermore, we can also consider constructing regression models for the recognition of different traffic rates and patterns, the recognition of multiple flows, or the use of semi-supervised learning models to better cope with unseen data. We will discuss this in more detail in Chapter 7.

6.4.4 Prototype demonstrator

So far, the performance of our models has been evaluated on infrastructures and servers with considerable amounts of resources. However, we will now demonstrate how our models can be used in a real-life setting. For this, we make use of the setup created for the real-life data collection in Section 6.4.3 and illustrated in Figure 6.9. The used hardware and the connections between the different devices are identical. However, instead of using the Intel NUC at the receiving side (at the right side of Figure 6.9) only for storage, we will now also implement the data adaptation steps and a classification model. As such, demonstrating the real-life applicability on more every-day devices. For this prototype, we implement the classification model for transmission rate, using time domain images (denote as CNN 5 before). Time domain images are chosen because they require less complex mathematical computations than STFT, making them faster to generate, which is important in a real-time setting. For the traffic flows, we select the generated IQ samples of four different scenarios with different characteristics. We selected traces with different values for the transmission rate, while other label values were chosen arbitrarily. As before, these IQ samples are transmitted using GNU Radio. The selected traffic classes and their characteristics are depicted in Table 6.6. The experiment was conducted in both an interference-free and an occupied channel in the 2.4 GHz frequency band.

The resulting confusion matrices for both runs can be seen in Figure 6.13. In total, 250 snapshots of 5 s per traffic stream were created for each experiment. The overall accuracy for the first experiment in the empty channel is 83.0 %. For the second experiment in the occupied channel, an accuracy of 66.5 % is achieved. From both confusion matrices, it is clear that the most miss-classifications oc-

Traffic class	Pattern	Protocol	Rate
Trace 0	High (75 %)	UDP	50 Mbps
Trace 1	Medium (50 %)	TCP	1 Mbps
Trace 2	Low (25 %)	UDP	100 Kbps
Trace 3	Medium (50 %)	TCP	10 Mbps

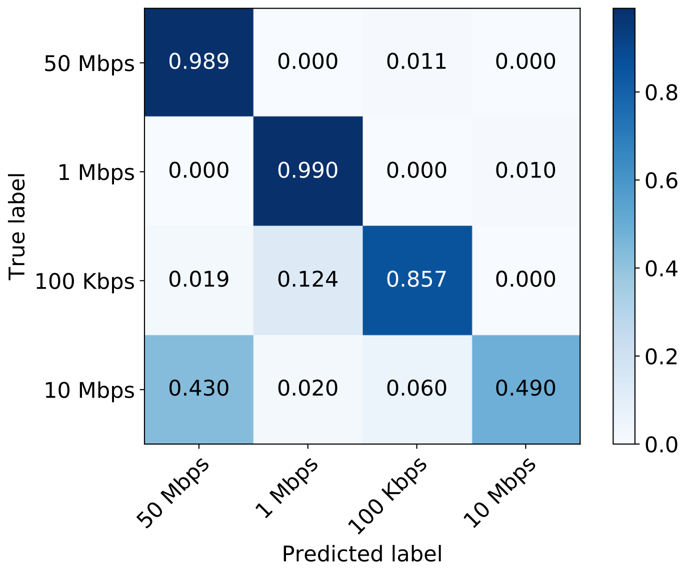
Table 6.6: Traffic classes and their characteristics for prototype.

curred for the 10 Mbps traffic (i.e., trace 3). This is similar to the results presented in the previous sections, where also 10 Mbps traffic was regularly classified as 50 Mbps traffic. Furthermore, we noticed that it took up to 5-6 s to convert the captured IQ samples to the images in time domain. This was especially the case for the 50 Mbps traffic, while the smaller 100 Kbps traffic took less than half of that time. This data adaption time can be decreased by implementing this feature directly in hardware or averaging IQ samples directly on the capturing interface, before converting them to images.

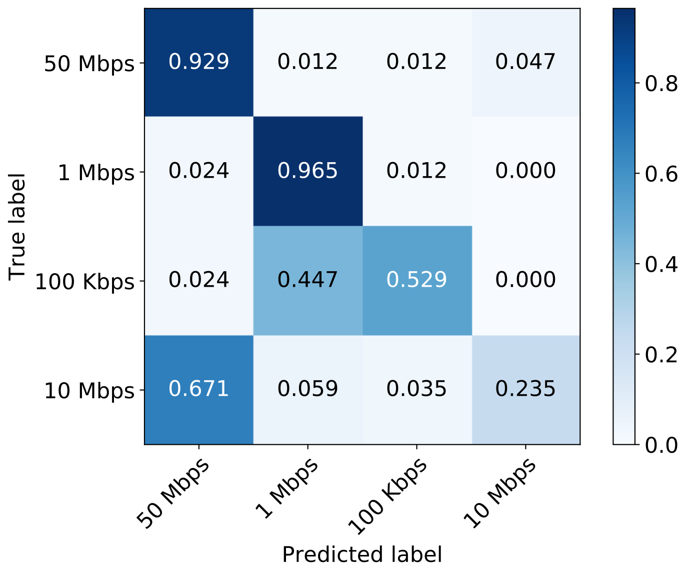
Finally, so far we have always reported the accuracy for all the models by using a single classification result. However, an application using the reported results could take the average result over a number of snapshots. This can, for instance, be done using a sliding window. We have tested this out for window sizes of 3, 5, and 7. Figure 6.14 shows the resulting accuracies across different window sizes for both experiments The best results are obtained when using an average over 5 samples, per traffic class: 90.0 % and 75.0 %, respectively for the experiments with an interference-free channel and an occupied channel. This means that there is an increase of, respectively, 7 % and 8.5 %. The resulting confusion matrices for this optimal window size are shown in Figure 6.15.

6.5 Conclusion

In this chapter, we have presented the first approach, to the best of our knowledge, that performs traffic recognition on spectral data. This strongly contrasts more intrusive traditional methods based on DPI or packet traces, as the listening device does not need to be part of the network and modern privacy requirements are respected. In particular, we have presented CNN models that are capable of performing three classification tasks: differentiate between TCP and UDP traffic, recognize constant and burst traffic with different duty cycles, and identify different transmission rates. Furthermore, we have successfully explored principles behind domain randomization to generate large amounts of synthetic (input) data. When evaluating the models with the test dataset with synthetic data, accuracies of more than 96 % are obtained. Using time-domain data allows for even higher accuracies of more than 99 %. Finally, we have performed a validation with real-



(a) interference-free channel



(b) Occupied channel

Figure 6.13: Normalized confusion matrices for the prototype

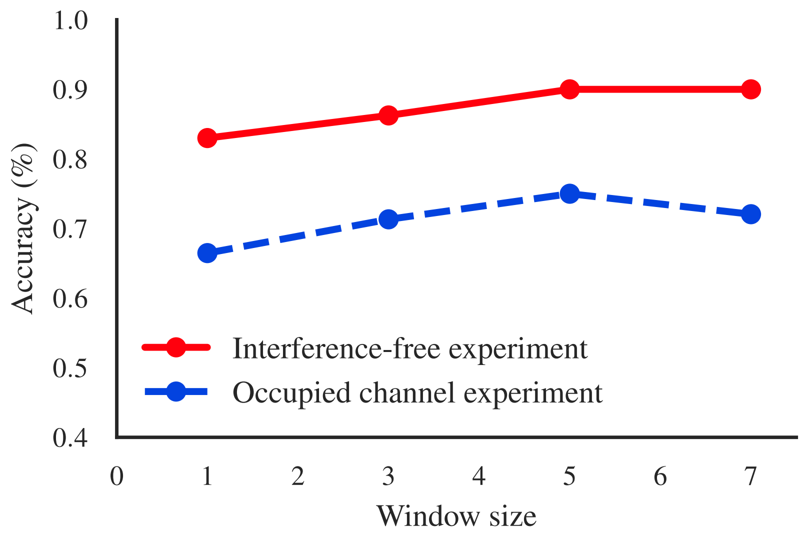
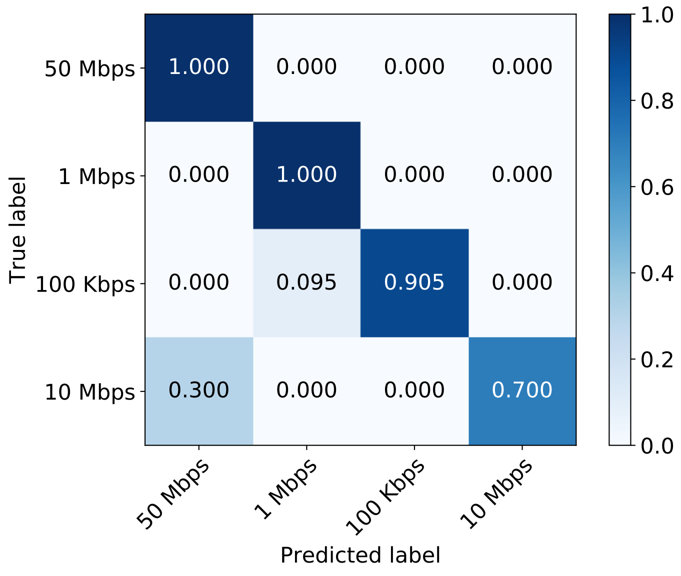
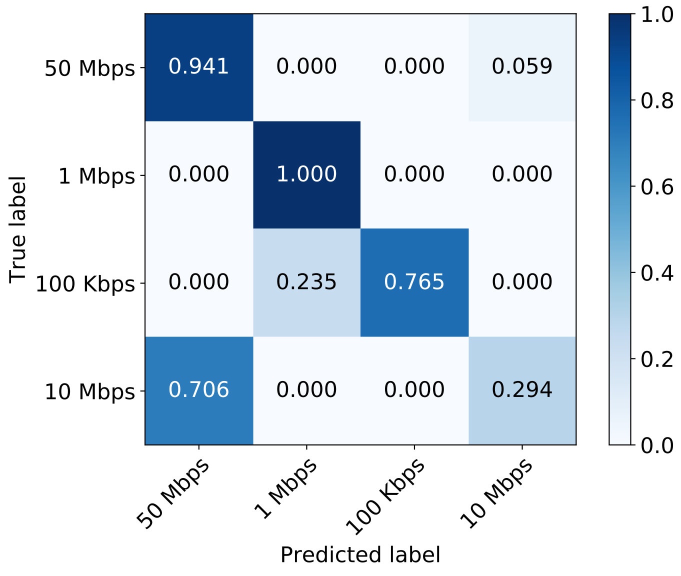


Figure 6.14: Overview of accucaries obtained using difference window sizes.

life data. Here we noticed that the overall accuracies are lower, although it is still possible to detect different transmission rates with an accuracy of 86.9 %. Overall, we have successfully explored the possibility of detecting traffic patterns at the spectral level and the use of synthetically generated training data. As such, a good starting point is provided for further research.



(a) interference-free channel



(b) Occupied channel

Figure 6.15: Normalized confusion matrices for the prototype using a combination of 5 snapshots

7 Conclusions and Perspectives

"My watch has ended"

–Jon Snow (Game of Thrones: Season 6 Episode 3, 2016)

In this dissertation, multiple contributions in the area of multi-technology network management are presented, in particular towards heterogeneous wireless networks. The proposed solutions either enable inter-technology management features (like handovers or load balancing), optimize the configuration of different wireless networks to increase network-wide throughput, or allow to detect traffic patterns of neighboring networks. This concluding chapter summarizes how this dissertation addresses the multiple problems identified in the area of wireless network management and verifies if the hypothesis and research questions have, respectively, been validated and answered appropriately. Furthermore, we conclude by identifying different interesting challenges or research topics that can further be addressed by the research community.

7.1 Review of problem statements

Each of the solutions proposed in the previous chapters addresses one of the problem statements presented in Chapter 1 as follows:

1. **Communication technologies operate fully independently of each other.** This is mainly due to the design of the lower layers of the OSI network stack. As such, this results in the inefficient use of wireless resources and connection interruptions or losses, which both have a dramatic effect on the

performance of applications and the end-user experience. Chapter 3 introduces the ORCHESTRA framework for inter-technology management. The framework consists of two key parts: the VMAC layer and a centralized controller. The VMAC offers a single connection point to the upper layers, while transparently bonding over the underlying network technologies. On the other hand, the controller introduces a single point of control and coordination across the entire network. Key features are seamless inter-technology handovers, packet-level load balancing, and duplication. In contrast to many existing approaches, the ORCHESTRA framework is completely independent towards upper (e.g., applications or transport protocols) and lower layers (i.e., technologies), allowing the deployment of the framework in a multitude of applications domains (e.g., LANs, backhauling networks, or satellite networks). An in-depth evaluation, using a real-life prototype setup, demonstrates that the presented features work as intended, and behave similarly or better than the default industry solution MPTCP. Table 7.1 revisits the original Table 2.1 from Chapter 2 and compares the proposed ORCHESTRA solution to the most important existing solutions. It is clear that the main novelty of ORCHESTRA lays in the combination of packet-level control with network-wide coordination. MPTCP is the only solution that offers the same level of control, but does so exclusively between two endpoints and not globally. Due to this fine-grained control, ORCHESTRA can offer more advanced features like the packet-level load balancing, and the duplication of critical data. Furthermore, ORCHESTRA is not limited to certain technologies and application domains, in contrast to approaches like IEEE 1905.1 or LTE-LWA.

2. **Autonomous and real-time coordination, across the different devices in a network, is missing.** Therefore, despite supporting multiple technologies, devices tend to connect to the Internet using a single technology, based on predefined priorities. Instead, we have presented a dynamic flow management approach in Chapter 4. Based on real-time monitoring information, the approach aims to utilize the full capacity of the network and react in an autonomous fashion to the inevitable dynamic network changes and differences in traffic demands. This becomes increasingly more important as the QoS and bandwidth requirements of modern services grow. In particular, we compare two different mathematical programming formulations that aim to maximize the network-wide throughput by dynamically rerouting the different traffic flows across all the available connections and network paths. The approach can be deployed on top of different management frameworks, especially on top of the proposed ORCHESTRA framework. A series of NS-3 based evaluations show that, on average, a network-wide throughput increase of 20 % can be obtained, while a small-scale prototype shows the applicability of the approach in real-life.

3. **Tailored management systems, that account for mobility and the ever-growing set of devices, are lacking.** This is because uneven load distribu-

Features	**IEEE 802.21**	**IEEE 1905.1**	**SDN-based**	**LTE-LWA**	**MPTCP**	**ORCHESTRA**
Network domains	LAN-WAN	LAN	LAN	LAN-RAN	Any (end-to-end)	**Any**
Technologies	3GPP , Wi-Fi, IEEE 802.16,	Ethernet, HP, Wi-Fi, MoCA	Wi-Fi, 3GPP	Wi-Fi, LTE	All	**All**
Coordination	None	Global	Global	Local (within cell)	Between end-points	**Global**
Control-level	Flow-based	Flow-based	Flow-based	Flow-based	Packet-based (sub-flows)	**Packet-based**
Transport protocols	Any	Any	Any	Any	only TCP	**Any**
Backward compatibility	Yes	No	No	Yes	Yes	**Yes**
Vertical Handovers	Yes	Yes	Yes	Yes (within cell)	Yes (between sub-flows)	**Yes**
Needs client changes	Yes (standards)	Yes	No	Yes	Yes	**Yes**
Products available	No	Qualcomm Hy-fi	Odin, 5G EmPOWER, ...	Two planned deployments	Android, iOS, Tessares, ...	**No**

Table 7.1: Comparison of the ORCHESTRA solution to the state-of-the-art.

tions among the available wireless technologies or network infrastructures, lead to suboptimal and inefficient use of the wireless resources, causing (significant) QoS degradations. To this extent, we continue the work presented in Chapter 4 by the contributions stated in Chapter 5. We present an MILP formulation for the problem of load balancing devices across different network connection infrastructures (e.g., Wi-Fi APs or LTE base stations), while also optimally scheduling traffic flows across different technologies. The goal of increasing the network-wide throughput and the objective of only making decisions based on real-time monitoring information are maintained from the preceding work in Chapter 4. As the optimal mathematical programming formulation scales exponentially, two different heuristics are proposed. Thorough evaluations based on NS-3 simulations show that the greedy heuristic is capable of increasing the network-wide throughput by more than 100 % across a variety of scenarios. Furthermore, we also demonstrate the scalability of the heuristic up to scenarios with 10000 devices.

4. **The coexistence of neighboring technologies and networks is heavily being pressured.** This is due to the deployment of an increasing number of wireless technologies and networks at overlapping or neighboring physical locations and increasing traffic volumes. In Chapter 6, we explore the possibility of detecting traffic patterns directly in the wireless spectrum. Detecting such patterns allows to increase the amount of information available to management systems to optimize the networks under their control. For instance, when management systems are capable of detecting these interference-free periods, traffic can be offloaded to these channels or technologies during these free slots. In order to address the difficulties faced when constructing real-life datasets and to increase the overall robustness of the models, we present a data generation framework that can be used to acquire large amounts of synthetic training and validation data, according to the domain randomization principle. We present a CNN architecture that is shared all models that are, respectively, able to recognize TCP and UDP traffic, burst traffic with different duty cycles, and different transmission rates. We show that all models have an accuracy of above 96 %, when using synthetic datasets containing both time and time-frequency image representations. A validation with real-life data samples and a small-scale prototype implementation show the potential of traffic recognition and the use of domain randomization approaches in a wireless networking context.

7.2 Review of the hypothesis and research questions

In Chapter 1, we have stated the following hypothesis: **Intelligent and dynamic inter-technology network management is needed to support the ever-evolving heterogeneous wireless networks**. This hypothesis envisioned, in layman terms, that a user is only aware of the fact that its device is connected to the Internet, while the network takes care of all the underlying decision-making and provisioning of resources, as such offering improved QoS and user experience (i.e., connectivity as a service). In this dissertation, we have presented a number of contributions to facilitate this vision. As mentioned above, the ORCHESTRA framework solves the problem of managing different communication technologies in a fundamental and transparent manner. Additionally, the intelligence introduced in Chapters 4 and 5 can be deployed on top of the ORCHESTRA framework to optimize the network and significantly increase the network-wide throughput. This is accompanied by traffic recognition models, in Chapter 6, to increase the monitoring capabilities of network management frameworks. Overall, we have clearly shown how our contributions can increase the network management of, in particular, wireless networks. For instance, by network-wide coordination, seamless inter-technology handovers, decision-making based on real-time monitoring, etc. Furthermore, these contributions account already for the future growth in numbers and diversity in wireless networks. Among others, the ORCHESTRA framework is technology independent, can be used with future communication technologies, and can be deployed in different application domains. Similarly, the proposed flow scheduling and load balancing algorithms can scale along with the rising numbers of connected devices and technologies. Note that when devices become equipped with more and novel communication technologies, the algorithms can load balance traffic and devices across even more connection points and network routes. Overall, we can conclude that intelligent and dynamic inter-technology network management is indeed needed to support the ever-evolving heterogeneous wireless networks.

The research questions formulated in Section 1.4, are answered as follows:

1. **How can we enable seamless inter-technology management transparent to all actors?** The ORCHESTRA framework, presented in Chapter 3, introduces the VMAC layer to transparently bond different physical interfaces into one connection towards the upper layers. No changes to the underlying IEEE 802 technologies are needed as they are fully compliant to the OSI model. In contrast, the traditional 3GPP architecture used in existing LTE networks is not suitable for the fine-grained control offered by the VMAC layer. However, follow-up 3GPP releases have introduced mechanisms (i.e., LBO) and architectures (e.g., LTE-LWA) that do open opportunities for deployment of the VMAC layer. Furthermore, the packet-based load balancing and duplication features operate also transparent to the upper layer because of the reordering and duplication functionalities at the receiving VMAC. Since the VMAC takes care of handling, among others, DHCP and ARP interactions, the VMAC appears transparent to the network as well. Furthermore, the second component of the ORCHESTRA framework, the controller, makes sure that all actions are coordinated across the network, as such delivering the needed seamless inter-technology management. Finally, in order to facilitate the roll-out of the framework, we made sure that communication with non-ORCHESTRA devices or existing SDN solutions are, to some extent, also possible. This can be seen as transparency towards

legacy devices and solutions.

2. **Can intelligent routing of traffic streams significantly improve network performance?** In Chapter 4, we introduced an MILP formulation that can dynamically reroute traffic flows across existing paths in the networks. We target, in particular, environments with both wired and wireless links that can benefit from offloading between the different technologies. The presented decision-making logic operates on top of, for instance, the ORCHESTRA framework and is based strictly on real-time monitoring information. An important aspect is that the specific nature of wireless networks (e.g., the impact of competing stations and the dynamic maximum capacity) is taken into account. We also describe how it is possible to dynamically estimate the maximum capacity of wireless technologies, a method that has recently been adopted by commercial Wi-Fi management solutions. Thorough evaluations show that the presented approach can indeed improve network performance by on average 20 % across a variety of scenarios. These evaluations include dynamic scenarios, a scenario with link failure, and a deployment of the approach in a real-life prototype.

3. **Can the impact of mobility and the growing number of devices and technologies be countered by introducing intelligent load balancing?** Yes, intelligent load balancing can indeed aid in coping with mobility and an increasing number of devices and technologies. We show this in Chapter 5, where we introduce an optimal mathematical load balancing formulation and two heuristic load balancing approaches. All three algorithms focus on balancing, for each supported technology, the connected devices across the available infrastructure devices, and scheduling the flows across the different connections. The load balancing is done by considering the traffic rates of the different flows, the distance between the devices and the connection points, the network load, and the maximum rates supported by the different devices and technologies. This is in contrast to the default approaches, where devices tend to connect to the AP or base station with the best SNR, regardless of the network load or spectrum occupancy. We demonstrate that our load balancing approaches can increase the network-wide throughput up to 130 %. Furthermore, the greedy heuristic can calculate an improved network configuration for 10000 devices in less than 3 s.

4. **Can we detect traffic patterns of neighboring networks using spectral data?** Traditional traffic recognition approaches, such as DPI, operate on a packet-level. This, typically, requires that the listening or capturing device is connected to the corresponding network. In contrast, we have investigated the detection of traffic patterns at the spectrum level. To the best of our knowledge, this problem has not been studied before. In Chapter 6, we have presented three prediction models that, respectively, can discriminate between different transport protocols (TCP versus UDP), traffic patterns (constant versus 3 types of burst traffic with a duty cycle of, respec-

tively, 25, 50, or 75 %), and transmission rates (100 Kbps, 1 Mbps, 10 Mbps, or 50 Mbps). Furthermore, by using our proposed data generation framework, we generate two different datasets containing images that represent snapshots of the spectrum in either time or time-frequency domain. Our evaluation shows that both approaches have an accuracy of more than 96 % for all models upon validation with synthetically generated data. However, the models trained with the dataset with images in the time-frequency domain, obtained after performing STFT, offer an even higher accuracy. This difference is especially significant when validating with real-life data. Overall, we can say that traffic patterns can indeed be detected at the spectrum level.

7.3 Future perspectives

This dissertation proposes multiple solutions to improve multi-technology network management. As this is a challenging and ever-evolving area, we have identified some remaining challenges and propose some ideas for future research. We list these areas for future work according to the corresponding contributions and chapters in this dissertation. Note that we bundle the topics related to Chapter 4 and Chapter 5, as the latter is an extension of the work proposed in Chapter 4.

7.3.1 Seamless inter-technology network management

- **Improve packet-based load balancing under TCP traffic.** In Section 3.6, we have compared the performance of the ORCHESTRA solution to the MPTCP standard, which is the default solution used by industry these days. While for handovers and duplication, the ORCHESTRA framework significantly outperforms the MPTCP solution, the packet-based load balancing feature under the presence of TCP can be improved. The encountered problem is due to the differences in latency across different wireless links. To this extent, we introduced a packet reordering mechanism, to make sure that packets are delivered in order to the higher layer. However, the differences in latency and the corresponding differences in packet arrival times, cause TCP still to react and latency to increase. As such, future work should consider better ways of minimizing the impact in terms of delay and TCP behavior effects. One interesting approach to investigate is the use of ML models. These models can, for instance, predict the future arrival rate of packets during a certain interval in order to normalize the inter-packet arrival times at the network layer.
- **Cope with uncontrollable networks or technologies.** The ORCHESTRA framework, as presented in Chapter 3, enables various management features across different communication technologies. The framework makes the assumption that the underlying technologies are controllable. While, for instance, for a backhauling use case, this is likely the case, this is not always

true. For instance, if a network operator wants to offload network traffic from a (controlled) LAN to the cellular connection that is controlled by a telecommunication company. Some existing commercial SDN solutions use a tunnel from a router to a cloud-based instance as a workaround to this problem. An interesting research idea could be to explore a cloud-based controller and (receiving) VMAC, to which network traffic is rerouted, but without the need to deploy tunnels (and thus avoid their overhead). However, the main challenge is that it becomes hard to identify individual packets at the cloud instance after they have passed through different networks. A so-called hole punching approach could aid in coping with firewalls and Network Address Translation (NAT), as such allowing to identify individual packets, which is needed to utilize the full functionalities of the ORCHESTRA framework [183]. This option could also be employed to cope with commercial LTE solutions that maintain the use of GTP tunnels. Furthermore, as mentioned in Section 3.3.2.2, the placement of the VMAC at the infrastructure side of cellular networks should also be studied in more detail. It is essential that the VMAC is reachable from all different technologies, in order for split flows (because of the packet-based load balancing or duplication) to be merged again. One of the challenges is that different technologies are typically placed in different subnetworks. This makes it hard to identify individual flows, similar to difficulties experienced with GTP tunnels. Besides the previously mentioned hole punching approach, alternatives that can be considered are header rewriting (modification to the addresses) or the incorporation of additional information in the headers of the data packets.

- **Support for packet scheduling across different technologies.** The ORCHESTRA framework, and in particular, the VMAC, enables MAC-level packet-level control across different technologies. Current features that employ this level of control are the (packet-based) load balancing and duplication across different technologies. However, we could extend this by looking at packet-level scheduling across different technologies. On top of the existing medium access schemes in the technologies themselves (e.g., CSMA/CA for Wi-Fi), we could design a scheduling mechanism at the VMAC layer that can consider transmission across different technologies. This idea can be seen as Multi-Frequency Time-Division Multiple Access (MF-TDMA) and could help in minimizing the inference across technologies or networks that operate in the same frequencies or channels. Note that this research direction is related to MAC scheduling efforts within the area of CR.

- **Kernel-level implementation.** In Section 3.5, we have presented a prototype implementation of the ORCHESTRA framework using the Click modular router. This prototype has proven its worth in evaluating the performance of the framework and comparing it to MPTCP. However, to further study the performance and capabilities of the framework, a kernel implementation, especially of the VMAC, is preferred. This would allow studying the memory and CPU impact of the VMAC of, in particular, the reordering and deduplication functionalities. Furthermore, such a kernel implementation would allow to export the framework to other devices like smartphones or IoT devices. Finally, this would also encourage to use of the framework by other researchers or companies.

7.3.2 Load balancing algorithms

- **Other objective criteria.** All algorithms presented in Chapters 4 and 5, were designed to increase the network-wide throughput. The evaluation of the algorithms, respectively, in Sections 4.5 and 5.4, have shown that the proposed approaches indeed succeed in this goal. Future work could consider other criteria as well, such as energy consumption, latency, or different QoS classes. Energy consumption can, for instance, be taken into account through the offloading of all traffic flows from a specific connection or technology. As such, this interface can be put into a sleep mode, which would save energy. Naturally, this can only be done, if the topology of the network and the traffic requirements allow it. Furthermore, different technologies have also different energy consumptions, typically depending on parameters such as MCS, link quality, distance, etc. The algorithms can be extended to take this behavior into account. Additionally, taking into account different QoS classes can, among others, allow the assignment of the required bandwidth to priority flows. While the existing load balancing approaches can optimize the best effort traffic as much as possible.

- **Coping with high mobility environments.** The different algorithms presented in Chapter 5 have been evaluated in scenarios with different mobility parameters. In particular, we employed the Random Waypoint Model, with a random start position, and a uniformly random chosen speed between 0.3 and 0.7 $\frac{m}{s}$. However, the impact of high mobility environments, such as VANETs, has not been investigated. This could be an interesting area to further explore. A good starting point could be the consideration of vehicular scenarios in the NS-3 simulator, as presented by Katsaros [184]. Afterwards, real-time experiments could be conducted at the Smart Highway testbed [185].

- **Prediction of patterns.** The different algorithms proposed in this dissertation make decisions based strictly on real-time monitoring information that is provided by the underlying frameworks, such as, ORCHESTRA. This is an improvement to the current state-of-the-art that typically assumed full knowledge about certain network or traffic flow information. However, we could also further advance this by not only making decisions based on the live state of the network but also consider predictions of the future state of the network. Using ML techniques, we can try to predict future patterns in terms of network load or mobility. This, in turn, can be used to further improve the network configurations.

- **Applicability in IoT environments.** In this dissertation, we have mostly focused on evaluations in LANs. However, further validation of the developed algorithms in real-life settings can be interesting to explore. Especially, the deployment in IoT environments, like a smart city or industry 4.0 setting, seems promising. This can be combined with adding energy consumption objectives to the algorithms, as explained above.

7.3.3 Traffic recognition in the wireless spectrum

- **Improve traffic recognition models.** In Chapter 6, we have introduced three models that are able to detect different traffic patterns at the level of the wireless spectrum. However, because of the difference in accuracy between the validation with synthetic and real-life data, there is room for improvement. This can, for instance, be done by using a part of the real-life datasets to retrain the models. It is also possible to generate additional training data based on the real-life data captures, using Generative Adversarial Networks (GANs). Another option would be to investigate the performance of semi-supervised learning methods that can better cope with unseen data. Furthermore, in order to cope better with noise and interference, we can add a noise filter to the data adaptation process, after capturing real-life samples. Finally, in the evaluation with real data (cf. Section 6.4.3) we noticed that the model for detecting traffic patterns achieves the lowest accuracy. Since the goal of this classification model is to detect patterns in a time series, we could explore the use of, or a combination with, Bayesian methods [186].

- **Extend traffic recognition models.** We have currently created six CNN-based classification models using a supervised learning approach. However, the current models for the detection of transmission rates and traffic patterns (i.e., duty cycle) can only discriminate according to four predefined labels. As real-life traffic cannot be expected to fall precisely within these four classes, a better approach would be to consider the transmission rate and the duty cycle as parameters to be identified. To this extent, the performance of regression models can be investigated. Furthermore, our models are currently trained to detect the characteristics of only a single traffic flow. Future work should extend this to cope with the presence of multiple overlapping flows. Finally, we can also push the idea of traffic recognition to the limit, by exploring the possibility of identifying traffic classes (e.g., video or voice traffic) or detecting unique applications (e.g., Netflix or Skype).

- **Further exploration of data randomization.** In Chapter 6, we were the first to explore the use of the domain randomization technique within the context of wireless networks. In the data generation framework, we augment pure IQ samples with different levels of noise to randomize the channel conditions. However, we believe that the full power of data randomization techniques is worth to explore further. For instance, instead of adding noise levels that are still realistic, we can investigate the impact of adding completely unrealistic values. This in light of the techniques used in the area of robotics and autonomous vehicles, as mentioned in Section 6.3.1. Furthermore, we can also try to randomize other aspects, such as Wi-Fi related parameters (e.g., channel-width, frequencies, or MCS). Note that these ideas can also be explored elsewhere within the area of wireless networks. For instance, to train the aforementioned prediction models for network load or mobility behavior.

- **Utilization by network management algorithms.** The traffic recognition models presented in this dissertation, provide new insights into the occupancy of the wireless spectrum. This information can be used by network management algorithms to optimize the network usage and offload traffic to frequencies, channels, or slots where bandwidth is still available. To this extent, future work should consider the development of such algorithms in order to close the loop. A particularly interesting research direction, is the combination of the traffic recognition models with the cross-technology packet-level scheduling extension to the ORCHESTRA framework, as mentioned above in Section 7.3.1. Traffic recognition models can help in providing the information needed to appropriate assign packets to the available slots across different technologies. Furthermore, traffic recognition information might also be valuable to construct traffic prediction models.

References

[1] B. M. Leiner, V. G. Cerf, D. D. Clark, R. E. Kahn, L. Kleinrock, D. C. Lynch, J. Postel, L. G. Roberts, and S. Wolff. *A brief history of the Internet*. ACM SIGCOMM Computer Communication Review, 39(5):22–31, 2009.

[2] N. Gershenfeld, R. Krikorian, and D. Cohen. *The internet of things*. Scientific American, 291(4):76–81, 2004.

[3] Cisco. *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*, 2011. Available from: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf.

[4] Cisco. *Cisco Visual Networking Index: Forecast and Trends, 2017-2022*, 2017. Available from: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf.

[5] I. Analytics. *State of the IoT 2018: Number of IoT devices now at 7B Market accelerating*, 2018. Available from: https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[6] Statista. *Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)*, 2016. Available from: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[7] E. Perahia and R. Stacey. *Next generation wireless LANs: 802.11 n and 802.11 ac*. Cambridge university press, 2013.

[8] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. *Internet of Things (IoT): A vision, architectural elements, and future directions*. Future generation computer systems, 29(7):1645–1660, 2013.

[9] N. Alliance. *5G white paper*. Next generation mobile networks, white paper, pages 1–125, 2015.

[10] M. S. Afaqui, E. Garcia-Villegas, and E. Lopez-Aguilera. *IEEE 802.11ax: Challenges and Requirements for Future High Efficiency WiFi*. IEEE Wireless Communications, 24(3):130–137, 2017. doi:10.1109/MWC.2016.1600089WC.

[11] B. Bellalta. *IEEE 802.11 ax: High-efficiency WLANs*. IEEE Wireless Communications, 23(1):38–46, 2016.

[12] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang. *What will 5G be?* IEEE Journal on selected areas in communications, 32(6):1065–1082, 2014.

[13] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi. *Internet of Things (IoT) communication protocols*. In Information Technology (ICIT), 2017 8th International Conference on, pages 685–690. IEEE, 2017.

[14] S. Chen, J. Hu, Y. Shi, and L. Zhao. *LTE-V: A TD-LTE-based V2X solution for future vehicular network*. IEEE Internet of Things journal, 3(6):997–1005, 2016.

[15] I. F. Akyildiz, D. M. Gutierrez-Estevez, and E. C. Reyes. *The evolution to 4G cellular systems: LTE-Advanced*. Physical communication, 3(4):217–244, 2010.

[16] R. Trestian, I.-S. Comsa, and M. F. Tuysuz. *Seamless multimedia delivery within a heterogeneous wireless networks environment: Are we there yet?* IEEE Communications Surveys & Tutorials, 20(2):945–977, 2018.

[17] M. Chiang and T. Zhang. *Fog and IoT: An overview of research opportunities*. IEEE Internet of Things Journal, 3(6):854–864, 2016.

[18] V. Sagar, R. Chandramouli, and K. P. Subbalakshmi. *Software defined access for HetNets*. IEEE Communications Magazine, 54(1):84–89, 2016.

[19] F. M. Abinader, E. P. Almeida, F. S. Chaves, A. M. Cavalcante, R. D. Vieira, R. C. Paiva, A. M. Sobrinho, S. Choudhury, E. Tuomaala, K. Doppler, and V. A. Sousa. *Enabling the coexistence of LTE and Wi-Fi in unlicensed bands*. IEEE Communications Magazine, 52(11):54–61, 2014.

[20] O. Galinina, A. Pyattaev, S. Andreev, M. Dohler, and Y. Koucheryavy. *5G multi-RAT LTE-WiFi ultra-dense small cells: Performance dynamics, architecture, and trends*. IEEE Journal on Selected Areas in Communications, 33(6):1224–1240, 2015.

[21] M. Z. Hasan, H. Al-Rizzo, and F. Al-Turjman. *A survey on multipath routing protocols for QoS assurances in real-time wireless multimedia sensor networks*. IEEE Communications Surveys & Tutorials, 19(3):1424–1456, 2017.

[22] H. Zimmermann. *OSI Reference Model-The ISO Model of Architecture for Open Systems Interconnection*. IEEE Transactions on Communications, 28(4):425–432, 1980.

[23] R. Wang, H. Hu, and X. Yang. *Potentials and Challenges of C-RAN Supporting Multi-RATs Toward 5G Mobile Networks*. IEEE Access, 2:1187–1195, 2014. doi:10.1109/ACCESS.2014.2360555.

[24] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos. *Software-defined and virtualized future mobile and wireless networks: A survey*. Mobile Networks and Applications, 20(1):4–18, 2015.

[25] X. Yan, Y. A. Şekercioğlu, and S. Narayanan. *A survey of vertical handover decision algorithms in Fourth Generation heterogeneous wireless networks*. Computer networks, 54(11):1848–1863, 2010.

[26] B. Dezfouli, V. Esmaeelzadeh, J. Sheth, and M. Radi. *A review of software-defined WLANs: Architectures and central control mechanisms*. IEEE Communications Surveys & Tutorials, 2018.

[27] N. Zhang, S. Zhang, S. Wu, J. Ren, J. W. Mark, and X. Shen. *Beyond Coexistence: Traffic Steering in LTE Networks with Unlicensed Bands*. IEEE Wireless Communications, 23(6):40–46, 2016.

[28] R. G. Garroppo, L. Gazzarrini, S. Giordano, and L. Tavanti. *Experimental assessment of the coexistence of Wi-Fi, ZigBee, and Bluetooth devices*. In World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a, pages 1–9. IEEE, 2011.

[29] M. Ndiaye, G. Hancke, and A. Abu-Mahfouz. *Software defined networking for improved wireless sensor network management: A survey*. Sensors, 17(5):1031, 2017.

[30] Y. Bejerano, S.-J. Han, and L. E. Li. *Fairness and load balancing in wireless LANs using association control*. In Proceedings of the 10th annual international conference on Mobile computing and networking, pages 315–329. ACM, 2004.

[31] D. Macone, G. Oddi, A. Palo, and V. Suraci. *A dynamic load balancing algorithm for Quality of Service and mobility management in next generation home networks*. Telecommunication Systems, 53(3):265–283, 2013.

[32] O. Olvera-Irigoyen, A. Kortebi, and L. Toutain. *Available Bandwidth Probing for path selection in heterogeneous home Networks*. In IEEE Globecom Workshops (GC Wkshps), pages 492–497, 2012.

[33] A. M. Cavalcante, E. Almeida, R. D. Vieira, S. Choudhury, E. Tuomaala, K. Doppler, F. Chaves, R. C. Paiva, and F. Abinader. *Performance evaluation of LTE and Wi-Fi coexistence in unlicensed bands*. In 2013 IEEE 77th Vehicular Technology Conference (VTC Spring), pages 1–6. IEEE, 2013.

[34] F. I. Di Piazza, S. Mangione, and I. Tinnirello. *On the effects of transmit power control on the energy consumption of WiFi network cards*. In International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness, pages 463–475. Springer, 2009.

[35] S. Rayanchu, A. Patro, and S. Banerjee. *Airshark: detecting non-WiFi RF devices using commodity WiFi hardware*. In Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference, pages 137–154. ACM, 2011.

[36] S. K. Sharma, T. E. Bogale, S. Chatzinotas, B. Ottersten, L. B. Le, and X. Wang. *Cognitive radio techniques under practical imperfections: A survey*. IEEE communications surveys and tutorials, 2015.

[37] S. Bayhan and A. Zubow. *Optimal mapping of stations to access points in enterprise wireless local area networks*. In Proceedings of the 20th ACM International Conference on Modelling, Analysis and Simulation of Wireless and Mobile Systems, pages 9–18. ACM, 2017.

[38] S. Palm. *Home networks: From bits to gigabits: Lessons learned from the Evolution of Home Networking*. IEEE Consumer Electronics Magazine, 1(3):29–35, 2012.

[39] P. Gallo, K. Kosek-Szott, S. Szott, and I. Tinnirello. *SDN@ home: A method for controlling future wireless home networks*. IEEE Communications Magazine, 54(5):123–131, 2016.

[40] A. De La Oliva, A. Banchs, I. Soto, T. Melia, and A. Vidal. *An overview of IEEE 802.21: media-independent handover services*. IEEE Wireless Communications, 15(4):96–103, 2008.

[41] K. Taniuchi, Y. Ohba, V. Fajardo, S. Das, M. Tauil, Y.-H. Cheng, A. Dutta, D. Baker, M. Yajnik, and D. Famolari. *IEEE 802.21: Media independent handover: Features, applicability, and realization*. IEEE Communications Magazine, 47(1):112–120, 2009.

[42] M.-S. Chiang, C.-M. Huang, P. B. Chau, S. Xu, H. Zhou, and D. Ren. *A forward fast media independent handover control scheme for Proxy Mobile IPv6 (FFMIH-PMIPv6) over heterogeneous wireless mobile network*. Telecommunication Systems, 65(4):699–715, 2017.

[43] V. Sharma, J. Kim, S. Kwon, I. You, and F.-Y. Leu. *An overview of 802.21 a-2012 and its incorporation into iot-fog networks using osmotic framework*. In International Conference on Internet of Things as a Service, pages 64–72. Springer, 2017.

[44] R. Marin-Lopez, F. Bernal-Hidalgo, S. Das, L. Chen, and Y. Ohba. *A new Standard for securing media independent handover: IEEE 802.21 A*. IEEE Wireless Communications, 20(6):82–90, 2013.

[45] IEEE Std. 1905.1-2013. *IEEE Standard for Convergent Digital Home Network for Heterogeneous Technologies*, 2013.

[46] A. Crabtree, R. Mortier, T. Rodden, and P. Tolmie. *Unremarkable networking: the home network as a part of everyday life*. In the Designing Interactive Systems Conference, pages 554–563, 2012.

[47] J.-P. Javaudin, M. Bellec, D. Varoutas, and V. Suraci. *OMEGA ICT project: Towards convergent Gigabit home networks*. In 19th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2008.

[48] T. Meyer, P. Langendörfer, M. Bahr, V. Suraci, S. Nowak, and R. Jennen. *An inter-MAC architecture for heterogeneous gigabit home networks*. In 20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2009.

[49] D. Macone, G. Oddi, A. Palo, and V. Suraci. *A dynamic load balancing algorithm for Quality of Service and mobility management in next generation home networks*. Telecommunication Systems, 53(3):265–283, 2013.

[50] K. Xu, X. Wang, W. Wei, H. Song, and B. Mao. *Toward software defined smart home*. IEEE Communications Magazine, 54(5):116–122, 2016.

[51] N. Soetens, J. Famaey, M. Verstappen, and S. Latré. *SDN-based management of heterogeneous home networks*. In 11th International Conference on Network and Service Management (CNSM), pages 402–405, 2015.

[52] A. Shalimov, D. Zuikov, D. Zimarina, V. Pashkov, and R. Smeliansky. *Advanced study of SDN/OpenFlow controllers*. Proceedings of the 9th Central & Eastern European Software Engineering Conference in Russia on - CEE-SECR '13, pages 1–6, 2013.

[53] L. Sequeira, J. L. de la Cruz, J. Ruiz-Mas, J. Saldana, J. Fernandez-Navajas, and J. Almodovar. *Building an SDN Enterprise WLAN Based on Virtual APs*. IEEE Communications Letters, 21(2):374–377, 2017.

[54] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao. *Towards programmable enterprise WLANS with Odin*. In Proceedings of the first workshop on Hot topics in software defined networks, pages 115–120. ACM, 2012.

[55] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed. *Programming Abstractions for Software-Defined Wireless Networks*. IEEE Transactions on Network and Service Management, 12(2):146–162, 2015.

[56] *5G-EmPOWER*. Available from: https://5g-empower.io/.

[57] H. Moura, G. V. Bessa, M. A. Vieira, and D. F. Macedo. *Ethanol: Software defined networking for 802.11 wireless networks*. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 388–396. IEEE, 2015.

[58] V. Sivaraman, T. Moors, H. Habibi Gharakheili, D. Ong, J. Matthews, and C. Russell. *Virtualizing the access network via open APIs*. In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, pages 31–42. ACM, 2013.

[59] *Wi-5: What to do with the Wi-Fi wild west*. Available from: http://www.wi5.eu.

[60] A. Mukherjee, J.-F. Cheng, S. Falahati, H. Koorapaty, D. H. Kang, R. Karaki, L. Falconetti, and D. Larsson. *Licensed-Assisted Access LTE: coexistence with IEEE 802.11 and the evolution toward 5G*. IEEE Communications Magazine, 54(6):50–57, 2016.

[61] C. Hoymann, D. Astely, M. Stattin, G. Wikstrom, J.-F. Cheng, A. Hoglund, M. Frenne, R. Blasco, J. Huschke, and F. Gunnarsson. *LTE release 14 outlook*. IEEE Communications Magazine, 54(6):44–49, 2016.

[62] D. Laselva, D. Lopez-Perez, M. Rinne, and T. Henttonen. *3GPP LTE-WLAN Aggregation Technologies: Functionalities and Performance Comparison*. IEEE Communications Magazine, 56(3):195–203, 2018.

[63] X. Wang, S. Mao, and M. X. Gong. *A survey of LTE Wi-Fi coexistence in unlicensed bands*. GetMobile: Mobile Computing and Communications, 20(3):17–23, 2017.

[64] P. Nuggehalli. *LTE-WLAN aggregation [Industry Perspectives]*. IEEE Wireless Communications, 23(4):4–6, 2016.

[65] P. Sharma, A. Brahmakshatriya, T. V. Pasca S., B. R. Tamma, and A. Franklin. *LWIR: LTE-WLAN Integration at RLC Layer with Virtual WLAN Scheduler for Efficient Aggregation*. In 2016 IEEE Global Communications Conference (GLOBECOM), pages 1–6, 2016.

[66] Y.-B. Lin, Y.-J. Shih, and P.-W. Chao. *Design and Implementation of LTE RRM With Switched LWA Policies*. IEEE Transactions on Vehicular Technology, 67(2):1053–1062, 2018.

[67] G. mobile Suppliers Association (GSA). *LTE in Unlicensed Spectrum: Trials, Deployments and Devices*, 2018. Available from: https://www.sata-sec.net/downloads/GSA/180117-GSA-Unlicensed-spectrum-report-Jan-2018.pdf.

[68] D. Chambers. *MulteFire lights up the path for universal wireless service*. Technical Report May, 2016.

[69] C. Rosa, M. Kuusela, F. Frederiksen, and K. I. Pedersen. *Standalone LTE in Unlicensed Spectrum: Radio Challenges, Solutions, and Performance of MulteFire*. IEEE Communications Magazine, 56(10):170–177, 2018.

[70] 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 communications magazine, 55(6):64–71, 2017.

[71] S. Parkvall, E. Dahlman, A. Furuskar, and M. Frenne. *NR: The new 5G radio access technology*. IEEE Communications Standards Magazine, 1(4):24–30, 2017.

[72] M. Networks. *Truffle - Broadband Bonding Appliance*. Available from: https://www.mushroomnetworks.com/truffle/.

[73] Peplink. *Multi-WAN Internet Load Balancer*. Available from: https://www.peplink.com/technology/internet-load-balancing/.

[74] C. Paasch, S. Ferlin, O. Alay, and O. Bonaventure. *Experimental evaluation of multipath TCP schedulers*. In Proceedings of the 2014 ACM SIGCOMM workshop on Capacity sharing workshop - CSWS '14, pages 27–32, 2014.

[75] A. Ford, C. Raiciu, M. Handley, and O. Bonaventure. *TCP Extensions for Multipath Operation with Multiple Addresses*. RFC 6824, RFC Editor, January 2013. Available from: http://www.rfc-editor.org/rfc/rfc6824.txt.

[76] Q. De Coninck, M. Baerts, B. Hesmans, and O. Bonaventure. *A first analysis of multipath TCP on smartphones*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 9631(September 2015):57–69, 2016.

[77] J. Kellokoski. *Real-life multipath TCP based make-before-break vertical handover*. In 2013 IEEE Symposium on Computers and Communications (ISCC), pages 000252–000256. IEEE, 2013.

[78] C. Paasch, G. Detal, F. Duchene, C. Raiciu, and O. Bonaventure. *Exploring mobile/WiFi handover with multipath TCP*. In Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design, pages 31–36. ACM, 2012.

[79] K. W. Choi, Y. S. Cho, J. W. Lee, S. M. Cho, J. Choi, et al. *Optimal load balancing scheduler for MPTCP-based bandwidth aggregation in heterogeneous wireless environments*. Computer Communications, 112:116–130, 2017.

[80] C. Paasch, R. Khalili, and O. Bonaventure. *On the benefits of applying experimental design to improve multipath TCP*. In Proceedings of the ninth ACM conference on Emerging networking experiments and technologies, pages 393–398. ACM, 2013.

[81] S. H. Baidya and R. Prakash. *Improving the performance of multipath TCP over heterogeneous paths using slow path adaptation*. In 2014 IEEE International Conference on Communications (ICC), pages 3222–3227. IEEE, 2014.

[82] S. C. Nguyen and T. M. T. Nguyen. *Evaluation of multipath TCP load sharing with coupled congestion control option in heterogeneous networks*. In Global Information Infrastructure Symposium-GIIS 2011, pages 1–5. IEEE, 2011.

[83] R. Khalili, N. Gast, M. Popovic, and J.-Y. Le Boudec. *MPTCP is not pareto-optimal: performance issues and a possible solution*. IEEE/ACM Transactions on Networking (ToN), 21(5):1651–1665, 2013.

[84] F. Rebecchi, M. D. De Amorim, V. Conan, A. Passarella, R. Bruno, and M. Conti. *Data offloading techniques in cellular networks: A survey*. IEEE Communications Surveys and Tutorials, 17(2):580–603, 2015.

[85] Tessares. *Hybrid Access Networks with MPTCP*. Available from: https://www.tessares.net/.

[86] O. Bonaventure. *Multipath TCP inside the beast*. Available from: http://blog.multipath-tcp.org/blog/html/2018/12/12/multipath_tcp_inside_the_beast.html.

[87] D. Lee, B. E. Carpenter, and N. Brownlee. *Media streaming observations: Trends in udp to tcp ratio*. International Journal on Advances in Systems and Measurements, 3(3-4), 2010.

[88] D. Murray, T. Koziniec, S. Zander, M. Dixon, and P. Koutsakis. *An analysis of changing enterprise network traffic characteristics*. In 2017 23rd Asia-Pacific Conference on Communications (APCC), pages 1–6. IEEE, 2017.

[89] Q. De Coninck and O. Bonaventure. *Multipath quic: Design and evaluation*. In Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, pages 160–166. ACM, 2017.

[90] J. W. Stewart III. *BGP4: inter-domain routing in the Internet*. Addison-Wesley Longman Publishing Co., Inc., 1998.

[91] Y. Rekhter, T. Li, and S. Hares. *A Border Gateway Protocol 4 (BGP-4)*. Technical report, Internet Engineering Task Force, 2005. Available from: https://www.rfc-editor.org/rfc/pdfrfc/rfc4271.txt.pdf.

[92] D. Meyer, L. Zhang, and K. Fall. *Report from the IAB Workshop on Routing and Addressing*. Technical report, Internet Engineering Task Force, 2007. Available from: https://www.rfc-editor.org/rfc/pdfrfc/rfc4984.txt.pdf.

[93] R. Shacham, H. Schulzrinne, S. Thakolsri, and W. Kellerer. *Session Initiation Protocol (SIP) Session Mobility*. Technical report, Internet Engineering Task Force, 2009. Available from: https://www.rfc-editor.org/info/rfc5631.

[94] M. Wernersson, S. Wanstedt, and P. Synnergren. *Effects of QoS scheduling strategies on performance of mixed services over LTE*. In 2007 IEEE 18th International Symposium on Personal, Indoor and Mobile Radio Communications, pages 1–5. IEEE, 2007.

[95] Apple. *About Wi-Fi Assist*. Available from: https://support.apple.com/en-us/HT205296.

[96] Kernel.org. *Linux Ethernet Bonding Driver HOWTO*, 2011. Available from: https://www.kernel.org/doc/Documentation/networking/bonding.txt.

[97] Interserver.net. *What is Network Bonding? Types of Network Bonding*, 2016. Available from: https://www.interserver.net/tips/kb/network-bonding-types-network-bonding/.

[98] S. Sahaly and P. Christin. *Inter-MAC forwarding and load balancing per flow*. In 20th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, pages 1–4, 2009.

[99] G. Oddi, A. Pietrabissa, F. D. Priscoli, and V. Suraci. *A decentralized load balancing algorithm for heterogeneous wireless access networks*. In World Telecommunications Congress, pages 1–6, 2014.

[100] O. Bouchet, A. Kortebi, and M. Boucher. *Inter-MAC green path selection for heterogeneous networks*. In IEEE Globecom Workshops (GC Wkshps), pages 487–491, 2012.

[101] A. Kortebi and O. Bouchet. *Performance evaluation of inter-mac green path selection protocol*. In 12th Annual IEEE Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET), pages 42–48, 2013.

[102] L.-H. Yen, J.-J. Li, and C.-M. Lin. *Stability and fairness of AP selection games in IEEE 802.11 access networks*. IEEE Transactions on Vehicular Technology, 60(3):1150–1160, 2011.

[103] I. Malanchini, M. Cesana, and N. Gatti. *Network selection and resource allocation games for wireless access networks*. IEEE Transactions on Mobile Computing, 12(12):2427–2440, 2013.

[104] L. Yang, Y. Cui, H. Tang, and S. Xiao. *Demand-aware load balancing in wireless lans using association control*. In 2015 IEEE Global Communications Conference (GLOBECOM), pages 1–6. IEEE, 2015.

[105] E. Coronado, R. Riggio, J. Villalón, and A. Garrido. *Wi-balance: Channel-aware user association in software-defined Wi-Fi networks*. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pages 1–9. IEEE, 2018.

[106] M. Zekri, B. Jouaber, and D. Zeghlache. *A review on mobility management and vertical handover solutions over heterogeneous wireless networks*. Computer Communications, 35(17):2055–2068, 2012.

[107] G. Gódor, Z. Jakó, Á. Knapp, and S. Imre. *A survey of handover management in LTE-based multi-tier femtocell networks: Requirements, challenges and solutions*. Computer Networks, 76:17–41, 2015.

[108] J. G. Andrews, S. Singh, Q. Ye, X. Lin, and H. S. Dhillon. *An overview of load balancing in HetNets: Old myths and open problems*. IEEE Wireless Communications, 21(2):18–25, 2014.

[109] D. Liu, L. Wang, Y. Chen, M. Elkashlan, K.-K. Wong, R. Schober, and L. Hanzo. *User association in 5G networks: A survey and an outlook*. IEEE Communications Surveys & Tutorials, 18(2):1018–1044, 2016.

[110] P. Coucheney, C. Touati, and B. Gaujal. *Fair and efficient user-network association algorithm for multi-technology wireless networks*. In IEEE INFOCOM 2009, pages 2811–2815. IEEE, 2009.

[111] Q. Ye, B. Rong, Y. Chen, M. Al-Shalash, C. Caramanis, and J. G. Andrews. *User association for load balancing in heterogeneous cellular networks*. IEEE Transactions on Wireless Communications, 12(6):2706–2716, 2013.

[112] D. Harutyunyan, S. Herle, D. Maradin, G. Agapiu, and R. Riggio. *Traffic-aware user association in heterogeneous LTE/WiFi radio access networks*. In NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, pages 1–8. IEEE, 2018.

[113] A. Alizadeh and M. Vu. *Load Balancing User Association in Millimeter Wave MIMO Networks*. IEEE Transactions on Wireless Communications, 18(6):2932–2945, 2019.

[114] B. Ng, A. Deng, Y. Qu, and W. K. Seah. *Changeover prediction model for improving handover support in campus area WLAN*. In Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP, pages 265–272. IEEE, 2016.

[115] S. Fernandes and A. Karmouch. *Vertical mobility management architectures in wireless networks: A comprehensive survey and future directions*. IEEE Communications Surveys & Tutorials, 14(1):45–63, 2012.

[116] Z. A. Qazi, J. Lee, T. Jin, G. Bellala, M. Arndt, and G. Noubir. *Application-awareness in SDN*. ACM SIGCOMM computer communication review, 43(4):487–488, 2013.

[117] T. AbuHmed, A. Mohaisen, and D. Nyang. *A survey on deep packet inspection for intrusion detection systems*. arXiv preprint arXiv:0803.0037, 2008.

[118] A. Dainotti, A. Pescape, and K. C. Claffy. *Issues and future directions in traffic classification*. IEEE network, 26(1):35–40, 2012.

[119] J. Sherry, C. Lan, R. A. Popa, and S. Ratnasamy. *Blindbox: Deep packet inspection over encrypted traffic*. ACM SIGCOMM Computer Communication Review, 45(4):213–226, 2015.

[120] M. Lotfollahi, R. S. H. Zade, M. J. Siavoshani, and M. Saberian. *Deep packet: A novel approach for encrypted traffic classification using deep learning*. arXiv preprint arXiv:1709.02656, 2017.

[121] F. Pacheco, E. Exposito, M. Gineste, C. Baudoin, and J. Aguilar. *Towards the deployment of Machine Learning solutions in network traffic classification: A systematic survey*. IEEE Communications Surveys & Tutorials, 2018.

[122] T. T. Nguyen and G. J. Armitage. *A survey of techniques for internet traffic classification using machine learning*. IEEE Communications Surveys and Tutorials, 10(1-4):56–76, 2008.

[123] J. Zhang, X. Chen, Y. Xiang, W. Zhou, and J. Wu. *Robust network traffic classification*. IEEE/ACM Transactions on Networking (TON), 23(4):1257–1270, 2015.

[124] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng. *Malware traffic classification using convolutional neural network for representation learning*. In 2017 International Conference on Information Networking (ICOIN), pages 712–717. IEEE, 2017.

[125] M. Shi, A. Laufer, Y. Bar-Ness, and W. Su. *Fourth order cumulants in distinguishing single carrier from OFDM signals*. In MILCOM 2008-2008 IEEE Military Communications Conference, pages 1–6. IEEE, 2008.

[126] E. Karami and O. A. Dobre. *Identification of SM-OFDM and AL-OFDM signals based on their second-order cyclostationarity*. IEEE transactions on vehicular technology, 64(3):942–953, 2015.

[127] E. Karami, O. A. Dobre, and N. Adnani. *Identification of GSM and LTE signals using their second-order cyclostationarity*. In 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, pages 1108–1112. IEEE, 2015.

[128] W. Liu, M. Kulin, T. Kazaz, A. Shahid, I. Moerman, and E. De Poorter. *Wireless technology recognition based on RSSI distribution at sub-Nyquist sampling rate for constrained devices*. Sensors, 17(9):2081, 2017.

[129] M. Schmidt, D. Block, and U. Meier. *Wireless interference identification with convolutional neural networks*. In 2017 IEEE 15th International Conference on Industrial Informatics (INDIN), pages 180–185. IEEE, 2017.

[130] S. Jeong, U. Lee, and S. C. Kim. *Spectrogram-Based Automatic Modulation Recognition Using Convolutional Neural Network*. In 2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN), pages 843–845. IEEE, 2018.

[131] S. Rajendran, W. Meert, D. Giustiniano, V. Lenders, and S. Pollin. *Deep learning models for wireless signal classification with distributed low-cost spectrum sensors*. IEEE Transactions on Cognitive Communications and Networking, 4(3):433–445, 2018.

[132] T. J. OShea, J. Corgan, and T. C. Clancy. *Convolutional radio modulation recognition networks*. In International conference on engineering applications of neural networks, pages 213–226. Springer, 2016.

[133] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter. *End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications*. IEEE Access, 6:18484–18501, 2018.

[134] S. Yi, H. Wang, W. Xue, X. Fan, L. Wang, J. Tian, and R. Matsukura. *Interference Source Identification for IEEE 802.15. 4 wireless Sensor Networks Using Deep Learning*. In 2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1–7. IEEE, 2018.

[135] E. Testi, E. Favarelli, and A. Giorgetti. *Machine Learning for User Traffic Classification in Wireless Systems*. In 2018 26th European Signal Processing Conference (EUSIPCO), pages 2040–2044. IEEE, 2018.

[136] P. Hintjens. *ZeroMQ: messaging for many applications*. O'Reilly Media, Inc., 2013.

[137] S.-Q. Lee and J.-u. Kim. *Local breakout of mobile access network traffic by mobile edge computing*. In 2016 International Conference on Information and Communication Technology Convergence (ICTC), pages 741–743, 2016. doi:10.1109/ICTC.2016.7763283.

[138] F. Giust, G. Verin, K. Antevski, J. Chou, Y. Fang, W. Featherstone, F. Fontes, D. Frydman, A. Li, A. Manzalini, et al. *MEC deployments in 4G and evolution towards 5G*. ETSI White Paper, 24:1–24, 2018.

[139] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella. *On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration*. IEEE Communications Surveys & Tutorials, 19(3):1657–1681, 2017. Available from: http://ieeexplore.ieee.org/document/7931566/, doi:10.1109/COMST.2017.2705720.

[140] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief. *A Survey on Mobile Edge Computing: The Communication Perspective*. IEEE Communications Surveys & Tutorials, 19(4):2322–2358, 2017. doi:10.1109/COMST.2017.2745201.

[141] F. Giust, V. Sciancalepore, D. Sabella, M. C. Filippou, S. Mangiante, W. Featherstone, and D. Munaretto. *Multi-access Edge Computing: The driver behind the wheel of 5G-connected cars*. IEEE Communications Standards Magazine, 2(3):66–73, 2018.

[142] G. T. 23.501. *Technical Specification Group Services and System Aspects: System Architecture for the 5G System, Stage 2 (Release 15),V1.3.0*, 2017.

[143] Y. Yu. *SDN-based Local breakout for mobile edge computing in radio access network*. In 2018 IEEE Wireless Communications and Networking Conference (WCNC), pages 1–6. IEEE, 2018.

[144] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, et al. *MEC in 5G networks*, 2018. Available from: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf.

[145] F. Khan. *Mobile Internet from the Heavens*. arXiv preprint arXiv:1508.02383, page 8, 2015.

[146] S. Xu, X.-w. Wang, and M. Huang. *Software-Defined Next-Generation Satellite Networks: Architecture, Challenges, and Solutions*. IEEE Access, 4(c):1–1, 2018.

[147] S. Al-Sarawi, M. Anbar, K. Alieyan, and M. Alzubaidi. *Internet of Things (IoT) communication protocols: Review*. In 2017 8th International Conference on Information Technology (ICIT), pages 685–690, 2017.

[148] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu. *Edge computing: Vision and challenges*. IEEE Internet of Things Journal, 3(5):637–646, 2016.

[149] P. A. Frangoudis, G. C. Polyzos, and V. P. Kemerlis. *Wireless community networks: An alternative approach for nomadic broadband network access*. IEEE Communications Magazine, 49(5):206–213, 2011.

[150] P. Bosch, J. Wyffels, B. Braem, and S. Latré. *How is your event Wi-Fi doing? Performance measurements of large-scale and dense IEEE 802.11n/ac networks*. In Proceedings of the IM 2017 - 2017 IFIP/IEEE International

Symposium on Integrated Network and Service Management, pages 701–707, 2017.

[151] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. *The click modular router*. ACM Transactions on Computer Systems, 18(3):263–297, 2000. doi:10.1145/354871.354874.

[152] *LEDE docs*. Available from: https://lede.readthedocs.io/en/latest/.

[153] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet. *OpenAirInterface: A flexible platform for 5G research*. ACM SIGCOMM Computer Communication Review, 44(5):33–38, 2014.

[154] *MultiPath TCP - Linux Kernel implementation*. Available from: https://multipath-tcp.org/pmwiki.php.

[155] I. Lopez, M. Aguado, C. Pinedo, and E. Jacob. *SCADA systems in the railway domain: enhancing reliability through Redundant MultipathTCP*. In 2015 IEEE 18th International Conference on Intelligent Transportation Systems, pages 2305–2310. IEEE, 2015.

[156] *Gurobi Optimization*. Available from: http://www.gurobi.com/.

[157] *CPLEX Optimizer*. Available from: https://www.ibm.com/analytics/cplex-optimizer.

[158] L. C. Coelho. *Linearization of the product of two variables*. Available from: https://www.leandro-coelho.com/linearization-product-variables/.

[159] *Wi-Fi Doctor: Keeping your WLAN healthy - White Paper - The Future Trust*, 2014. Available from: https://www.slideshare.net/TechnicolorCo/white-paper-wifi-doctor.

[160] T. R. Henderson, M. Lacage, G. F. Riley, C. Dowell, and J. Kopena. *Network simulations with the ns-3 simulator*. SIGCOMM demonstration, 14(14):527, 2008.

[161] A. T. NoteTN2224. *Best Practices for Creating and Deploying HTTP Live Streaming Media for Apple Devices*. Technical report, Apple, 2012. Available from: https://developer.apple.com/library/ios/technotes/tn2224/_index.html.

[162] Y. Donoso and R. Fabregat. *Multi-objective optimization in computer networks using metaheuristics*. CRC Press, 2016.

[163] H. Liu, H. Darabi, P. Banerjee, and J. Liu. *Survey of wireless indoor positioning techniques and systems*. IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 37(6):1067–1080, 2007.

[164] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[165] F. Tschopp, J. N. Martel, S. C. Turaga, M. Cook, and J. Funke. *Efficient convolutional neural networks for pixelwise classification on heterogeneous hardware systems*. In 2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI), pages 1225–1228. IEEE, 2016.

[166] Y. Chen, J. Wang, B. Zhu, M. Tang, and H. Lu. *Pixel-wise deep sequence learning for moving object detection*. IEEE Transactions on Circuits and Systems for Video Technology, 2017.

[167] R. Guo, J. Liu, N. Li, S. Liu, F. Chen, B. Cheng, J. Duan, X. Li, and C. Ma. *Pixel-Wise Classification Method for High Resolution Remote Sensing Imagery Using Deep Neural Networks*. ISPRS International Journal of Geo-Information, 7(3):110, 2018.

[168] Y. LeCun and Y. Bengio. *Convolutional Networks for Images, Speech, and Time Series*. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks, pages 255–258. MIT Press, 1998.

[169] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. *Deep learning for computer vision: A brief review*. Computational intelligence and neuroscience, 2018, 2018.

[170] X. Glorot, A. Bordes, and Y. Bengio. *Deep sparse rectifier neural networks*. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 315–323, 2011.

[171] G. E. Dahl, T. N. Sainath, and G. E. Hinton. *Improving deep neural networks for LVCSR using rectified linear units and dropout*. In 2013 IEEE international conference on acoustics, speech and signal processing, pages 8609–8613. IEEE, 2013.

[172] K. O'Shea and R. Nash. *An introduction to convolutional neural networks*. arXiv preprint arXiv:1511.08458, 2015.

[173] D. P. Kingma and J. Ba. *Adam: A method for stochastic optimization*. In International Conference on Learning Representations (ICLR), 2015.

[174] F. Chollet et al. *Keras*. Available from: https://keras.io.

[175] M. Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. Software available from tensorflow.org. Available from: https://www.tensorflow.org/.

[176] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. *Sim-to-real transfer of robotic control with dynamics randomization*. In 2018 IEEE International Conference on Robotics and Automation (ICRA), pages 1–8. IEEE, 2018.

[177] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. *Domain randomization for transferring deep neural networks from simulation to the real world*. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 23–30. IEEE, 2017.

[178] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield. *Training deep networks with synthetic data: Bridging the reality gap by domain randomization*. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 969–977, 2018.

[179] J. Tobin, L. Biewald, R. Duan, M. Andrychowicz, A. Handa, V. Kumar, B. McGrew, A. Ray, J. Schneider, P. Welinder, et al. *Domain randomization and generative models for robotic grasping*. In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3482–3489. IEEE, 2018.

[180] *Mathworks Products and Services*. Available from: https://www.mathworks.com/products.html.

[181] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter. *End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications*. IEEE Access, 6:18484–18501, 2018.

[182] *GNU Radio: The free & Open Software Radio Ecosystem*. Available from: https://www.gnuradio.org/.

[183] B. Ford, P. Srisuresh, and D. Kegel. *Peer-to-Peer Communication Across Network Address Translators*. In USENIX Annual Technical Conference, General Track, pages 179–192, 2005.

[184] K. Katsaros. *Vehicular Communication Simulations with NS-3*. Available from: http://www.nsnam.org/tutorials/consortium15/Vehicular-Comms.pptx.

[185] IDLab. *CityLab: a unique edge-computing and multi-wireless-technology smart cities testbed*. Available from: https://www.uantwerpen.be/en/research-groups/idlab/infrastructure/.

[186] C. Davidson-Pilon. *Bayesian Methods for Hackers: Probabilistic Programming and Bayesian Inference*. Addison-Wesley Data & Analytics Series, 2016.