UC Irvine

UC Irvine Electronic Theses and Dissertations

Title

Adaptive Communications for Intelligent and Autonomous Systems in the Urban Internet-of-Things (IoT)

Permalink

https://escholarship.org/uc/item/3k68r3pq

Author

Baidya, Sabur Hassan

Publication Date

2019

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA,
IRVINE

Adaptive Communications for Intelligent and Autonomous Systems in the Urban Internet-of-Things (IoT)

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Science

by

Sabur Hassan Baidya

Dissertation Committee:
Professor Marco Levorato, Chair
Professor Nikil Dutt
Professor Nalini Venkatasubramanian

2019

Portion of Chapter 3 © 2016 IEEE
Portion of Chapter 3,5 © 2017 IEEE
Portion of Chapter 3,4 © 2018 IEEE
Portion of Chapter 7 © 2018 ACM
Portion of Chapter 9 © 2019 IEEE
Portion of Chapter 9 © 2019 ACM
All other materials © 2019 Sabur Hassan Baidya

DEDICATION

I dedicate my PhD thesis...

...

...in loving memory of my father Md. Shahidullah, who unfortunately didn't stay in this world long enough to see his son completing PhD

...

...to my affectionate mother Roshenara Khatun who sacrificed her career and dedicated her whole life in supporting me and my siblings.

...

...and, to my beloved wife Fariah Asha Haque who has been a constant source of inspiration, motivation and support throughout this journey.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my advisor **Professor Marco Levorato** for his endless support, guidance and encouragements for my research and also his empathy and emotional support when I had personal loss during the course of my PhD. This dissertation wouldn't have been possible without his persistent help in exchanging and developing ideas, his continuous advice during the implementations and his immense help in writing and presenting the research papers. I feel blessed and fortunate to have him as a guide and mentor and couldn't ask for a better advisor.

I would like to thank the other members of my dissertation committee, **Professor Nikil Dutt** and **Professor Nalini Venkatasubramanian** for their valuable feedback and suggestions. Also, special thanks to my internship mentors Yan Chen from Huawei Research America lab and Prasanth Ananth from Nokia Bell Labs for their valuable guidance. During my PhD, I closely worked with my labmate Davide Callegaro and also shared eventful, memorable experiences of outdoor experiments with drones. I am also thankful to my labmates Zoheb, Peyman, Yoshi, Igor, Anas for valuable discussions, technical help and collaboration. Special thanks to Biswadip and Yunho from neighboring labs for their generous help, suggestions and constructive discussions. I am thankful to all the external collaborators including Aakanksha Chowdhery from Google Brain, Gowri Ramachandran and Professor Bhaskar Krishnamachari from University of Southern California for their help and advice.

My PhD research was supported in part by: DARPA grant HR00111910001, NSF grant IIS-1724331, and Donald Bren School of Information and Computer Sciences at UCI. Thanks to all of them for their kind support.

Finally, I want to thank my entire family for all their support, sacrifices and blessings. I am eternally indebted to my parents who in spite of all hardships in life prioritized education above everything and taught me discipline, diligence and perseverance. I am grateful for all the support I got from my elder brother, elder sisters, my twin brother, my brothers-in-law, sisters-in-law and parents-in-law, that helped me achieving this PhD. Last but not the least, I am forever grateful to **Fariah Asha Haque**, my wife and my best friend, who has been a pillar of support during all ups and downs in this journey. Her love and endless support helped me to get through difficult times when I lost my father in the middle of PhD and encouraged me to push forward to complete my PhD which was my father's dream.

CURRICULUM VITAE

Sabur Hassan Baidya

EDUCATION

University of California, Irvine — Aug'19
PhD in Computer Science — GPA: 3.95/4

University of Texas at Dallas — Aug.'13
MS in Computer Science (awarded Academic Excellence) — GPA: 3.96/4

West Bengal University of Technology
B.Tech in Electronics & Communication Engineering — Aug.'07
Class rank : 2 out of 65 students in ECE dept. — GPA: 8.93/10

PROFESSIONAL EXPERIENCE

Nokia Bell Labs, *Research Intern* — Jun.'17 - Sept.'17
Edge-Cloud Research — *Murray Hill, NJ*

Huawei Research Lab, *Research Intern* — Jun.'16 - Sept.'16
Network Virtualization Group — *Santa Clara, CA*

Cisco Systems, *Software Engineer* — Sept.'13 - Sept.'14
Software Routing Group for 3G/4G — *San Jose, CA*

BlackBerry Ltd., *Software Developer Intern* — Jan.'13 - May'13
Radio Applications R&D — *Irving, TX*

IBM, *Senior System Engineer* — Sept.'07 - Jun.'11
Telecom Group — *Noida, India*

ACADEMIC RESEARCH EXPERIENCE

Intelligent & Autonomous Systems Lab, UCI, CA — Sept.'14 - present
Graduate Research Assistant, Adviser: Dr. Marco Levorato — *Irvine, CA*

Distributed Systems Lab, UT Dallas, TX — Sept.'11 - Aug.'13
Research Student, Adviser: Dr. Ravi Prakash — *Dallas, TX*

WINLAB, Rutgers University, NJ — May'12 - Aug.'12
Research Intern, Adviser: Dr. Dipankar Raychaudhuri — *New Brunswick, NJ*

TEACHING EXPERIENCE

Graduate Teaching Assistant 2015 - 2016
University of California, Irvine *Irvine, CA*

Teaching Assistant 2011
University of Texas at Dallas *Dallas, TX*

HONORS & AWARDS

- **Student Travel Grant** offer for ACM SIGCOMM Conference. 2019
- **People's Choice Award**, Graduate Research Symposium, UCI. 2018
- **NSF Travel Grant** for ACM MobiHoc Conference. 2018
- **SIGMETRICS Travel Grant** for ACM SIGMETRICS Conference. 2018
- **Best Poster Award** in Computer Science Research Showcase, UCI. 2016
- **Third best poster award** in the Intern Research Showcase at Huawei Research Labs, Santa Clara, CA. 2016
- Stipend award for **Mentoring Excellence** at UC Irvine. 2015 - 2017
- **Graduate Fellowship** from Computer Science dept. of UC Irvine. 2014
- **Certificate of Academic Excellence**, Computer Science Department, University of Texas at Dallas. 2013
- Nominated for **'Golden Key International Honour Society'** by the University of Texas at Dallas for academic excellence. 2012
- 5th Place award in the workshop and competition on Cyber Security and ethical hacking at TexSAW in University of Texas at Dallas. 2011
- **IDB** scholarship for 4 years of undergraduate studies. 2003-2007

POSTER PUBLICATIONS

eBPF Filtering and Packet Processing [3rd Best Poster] **Sept 2016**
Intern Research Showcase, Huawei Research Lab, CA

Content-based Cognitive Interference Control for City Monitoring Applications in the Urban IoT [Best Poster] **June 2016**
Computer Science Dept, UC Irvine

Multihoming in Mobility First Internet Architecture **Aug 2012**
WINLAB Summer Research Program Open House, Rutgers University

BOOK CHAPTER

Urban IoT Edge Analytics **2018**
Fog Computing in the Internet of Things, Springer International Publishing

REFEREED JOURNAL PUBLICATIONS

Content-Aware Cognitive Interference Control for Urban IoT Systems **March 2018**
IEEE Transactions on Cognitive Communications and Networking

REFEREED CONFERENCE PUBLICATIONS

A Measurement Study on Edge Computing for Autonomous UAVs **Aug 2019**
ACM SIGCOMM Workshop on Autonomous Mobile AirGround Edge Computing, Systems, Networks, and Applications

FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot **Oct 2018**
ACM international conference on Modeling, analysis & simulation of wireless and mobile systems (MSWiM)

eBPF-based Content and Computation-aware Communication for Real-time Edge Computing **May 2018**
IEEE INFOCOM Workshop on Advances in Software Defined and Context-Aware Cognitive Networks

Robust Multi-Path Communications for UAVs in the Urban IoT **Apr 2018**
IEEE SECON Workshop on Communications, Data Processing and Control for Unmanned Autonomous Systems

Edge-assisted Content and Computation-Driven Dynamic Network Selection for Real-Time Services in the Urban IoT **May 2017**
IEEE INFOCOM Workshop on Advances in Software Defined and Context-Aware Cognitive Networks

Content-Based Interference Management for Video Transmission in D2D Communications Underlaying LTE **Jan 2017**
IEEE International Conference on Computing, Networking and Communications (ICNC)

Content-based Cognitive Interference Control for City Monitoring Applications in the Urban IoT **Dec 2016**
IEEE Global Communications Conference (GLOBECOM)

Improving the performance of Multipath TCP over Heterogeneous Paths using Slow Path Adaptation **June 2014**
IEEE International Conference on Communications (ICC)

ABSTRACT OF THE DISSERTATION

Adaptive Communications for Intelligent and Autonomous Systems
in the Urban Internet-of-Things (IoT)

By

Sabur Hassan Baidya

Doctor of Philosophy in Computer Science

University of California, Irvine, 2019

Professor Marco Levorato, Chair

Wireless communications for intelligent and autonomous systems in the urban Internet-of-Things (IoT) involve immense challenges, especially for time-sensitive and mission-critical applications and/or disruptive environment scenarios. Moreover, the enormous scale of wireless devices with heterogeneous technologies like LTE, Wi-Fi or other device-to-device (D2D) communications often share common wireless frequency spectrum and causes interference in the dense environment of urban IoT. The aforementioned scenarios result in high latency in the application, data losses in the wireless medium and often disruption in connections among communicating devices, thus failing to meet the quality of service or computation. Modeling the behaviors of the complex interactions of communication, computation and control in the real-world autonomous systems, e.g., unmanned aerial vehicles (UAV) in the dynamic environment is practically impossible. Our proposed approach learns the behavior of the application performance through the semantics of the data and also the cross-layer information from the network protocol stack, and formulates dynamic adaptive policies for improving the performance of the application and/or mission.

This dissertation proposes mitigation of these problems by realizing approaches, e.g., data-driven prediction in information constrained systems for dynamically selecting the best mode

of communication and computation for a given objective function, Stochastic optimization for cognitive interference control, Software defined networking protocols and network virtualization to facilitate the adaptive framework. For the prediction of quality of wireless networks in a multi-channel or multi-network environment, the notion of probes, resembling the data semantics is introduced to predict the available unused channels or networks. In complex scenarios involving more environmental dynamics, e.g., UAV based applications, prediction based on cross-layer information and physical parameters is used to facilitate dynamic network selection for the UAV mission. In case of interference from coexisting wireless devices sharing frequency spectrum, we propose a cognitive interference control framework where based on the relevance of the data content, i.e., if the data is part of a reference block or differentially encoded block, a Markov Decision Process (MDP) determines a transmission policy that achieves required data quality while optimizing the throughput of the secondary users. Additionally, in order to minimize the latency of decision making for data processing and/or forwarding in a multi-scale computing architecture, we incorporate Network Function Virtualization (NFV) employing inbuilt kernel feature, called extended Berkeley Packet Filter (eBPF) to parse the packets in the earlier phase of the packet reception in the kernel space. The packet processing and filtering functions implemented in the user space can be called by eBPF for fast packet filtering and forwarding on layer2 to minimize the socket buffer allocation; thus, saving both system resources and latency.

The proposed solutions are implemented both in simulation and on real-world systems, e.g., UAV equipped with sensors for data processing, and software defined radios (e.g. USRP B210, B200mini) for communication. This research also contributes to the development of an open-source based UAV-network simulator called "FlyNetSim" and open-source based NFV implementation with eBPF on Linux kernel, for free use in the research community.

Chapter 1

Introduction

1.1 Copyright Notice

Some material contained within this dissertation has been previously published and is used with permission.

- Copyright © 2019 IEEE. Reprinted with permission, from Davide Callegaro, Sabur Baidya, Gowri S. Ramachandran, Bhaskar Krishnamachari, and Marco Levorato, "Information Autonomy: Self-Adaptive Information Management for Edge-Assisted Autonomous UAV Systems" In IEEE MILCOM 2019-IEEE Military Communications Conference (MILCOM).

- Davide Callegaro, Sabur Baidya and Marco Levorato, "A Measurement Study onEdge Computing for Autonomous UAVs", in Proceedings of the ACM SIGCOMM 2019 Workshop on Mobile AirGround Edge Computing, Systems, Networks, and Applications (MAGESys) © 2019 ACM, Inc. `http://dx.doi.org/10.1145/3341568.3342109`. Reprinted by permission.

- Sabur Baidya, Zoheb Shaikh and Marco Levorato, "FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot", in Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWIM) © 2018 ACM, Inc. `http://dx.doi.org/10.1145/3242102.3242118`. Reprinted by permission.

- Copyright © 2018 IEEE. Reprinted with permission, from Sabur Baidya and Marco Levorato, "Content-Aware Cognitive Interference Control for Urban IoT Systems." in IEEE Transactions on Cognitive Communications and Networking, 4(3), pp.500-512.

- Copyright © 2018 IEEE. Reprinted with permission, from Sabur Baidya, Yan Chen, and Marco Levorato, "eBPF-based content and computation-aware communication for real-time edge computing." In IEEE INFOCOM 2018-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 865-870.

- Copyright © 2017 IEEE. Reprinted with permission, from Sabur Baidya, and Marco Levorato, "Edge-assisted content and computation-driven dynamic network selection for real-time services in the urban IoT." In 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 796-801.

- Copyright © 2017 IEEE. Reprinted with permission, from Sabur Baidya, and Marco Levorato, "Content-based interference management for video transmission in D2D communications underlaying LTE." In 2017 International Conference on Computing, Networking and Communications (ICNC), pp. 144-149.

- Copyright © 2016 IEEE. Reprinted with permission, from Sabur Baidya, and Marco Levorato, "Content-based cognitive interference control for city monitoring applications in the urban IoT." In 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1-6.

1.2 Motivation: Urban Internet-of-Things

Urban Internet-of-Things (IoT) involves interconnection and inter-operation of millions of devices to provide various multi-sensor based smart services in the urban environment. In near future majority of the population are expected to be living in urban areas [170] which necessitates a robust urban IoT infrastructure to support present and future applications. Moreover, the enormous amount of data flow over the network needs efficient transport mechanisms to support the corresponding computations and applications. The heterogeneity of the applications, computations and communication technologies make their coexistence more challenging. We study the impact of the dynamics of the urban environment on those heterogeneous components which motivates to formulate adaptive policies for computation and communications. Now, creating an urban IoT infrastructure for smart city has multifaceted challenges including socioeconomic, regulatory, demographic and many others. However, from technological perspective we focus on the Communication and computation challenges in the urban IoT.

1.2.1 Communication Challenges in Urban IoT

Supporting city-wide exchange of information in Urban Internet of Things (IoT) systems using existing communication infrastructures is extremely challenging especially when traditional services operate in the same network resource. Additionally, the most advanced Urban IoT services focus on real-time data processing, which shifts the perspective and goal of the Network when transporting data. In order to support the exchange of data streams, the urban IoT needs a robust network infrastructure support for communication over varying range. For small scale deployment of the sensors, a small wireless network, e.g., WiFi or Wireless Sensor Networks (WSN) may work. However, for long-range fast communication over wireless link, it may need LTE and 5G communications. The enormous volume of data

also needs to be transported across the city requiring availability of high bandwidth or intelligent means of transporting the traffic based on the dynamic requirement for computations. With the advent of cutting-edge technologies new applications are getting developed that necessitates high bandwidth and/or low latency communications - e.g., Virtual Reality (VR) and Augmented Reality (AR), Autonomous Vehicles, Vehicle-to-Vehicle (v2V) communications, real-time video surveillance with object detection, real-time live video streaming like First Person View (FPV) from an Unmanned Aerial Vehicle (UAV), real-time robotic controls etc. Apart from the latency and bandwidth requirement, handling the fast dynamics due to high speed mobility in vehicular communication and UAVs are extremely challenging. Another major problem in the dense environment e.g. cities is that enormous number of wireless devices coexist with heterogeneous wireless technologies sometimes sharing the same frequency spectrum causing interference in the wireless medium. Mitigating interference for real-time applications is one of the major challenges in the urban IoT.

1.2.2 Challenges for Intelligent & Autonomous Systems

The smart and intelligent services in urban IoT involves computations in different scales in different sites of the network. The end-devices equipped with multi-sensors usually comprises of components with low computational power, lower memory, light weight hardware. As a result on-board computations at the sensor devices may not be highly efficient in terms of completion time of a computation process, also the accuracy of computations. For example, in smart real-time surveillance systems, the camera captures the video in real-time and processing on-board at the camera may not provide high accuracy of object detection or takes longer time to compute the task, thus loosing the real-time nature of the application. similarly, if an UAV application involves tracking and following an object, then it needs to capture the video of the object, then compute and then make decision for control accordingly - the whole thing in real-time. Apart from the computational capability, there are constraints

of energy also as most of the sensor devices are operated with batteries. Hence, even if a computation is feasible to be done on-board it may consume lot of power that can limit the mission-oriented applications.

1.2.3 Multi-scale Computation Architecture

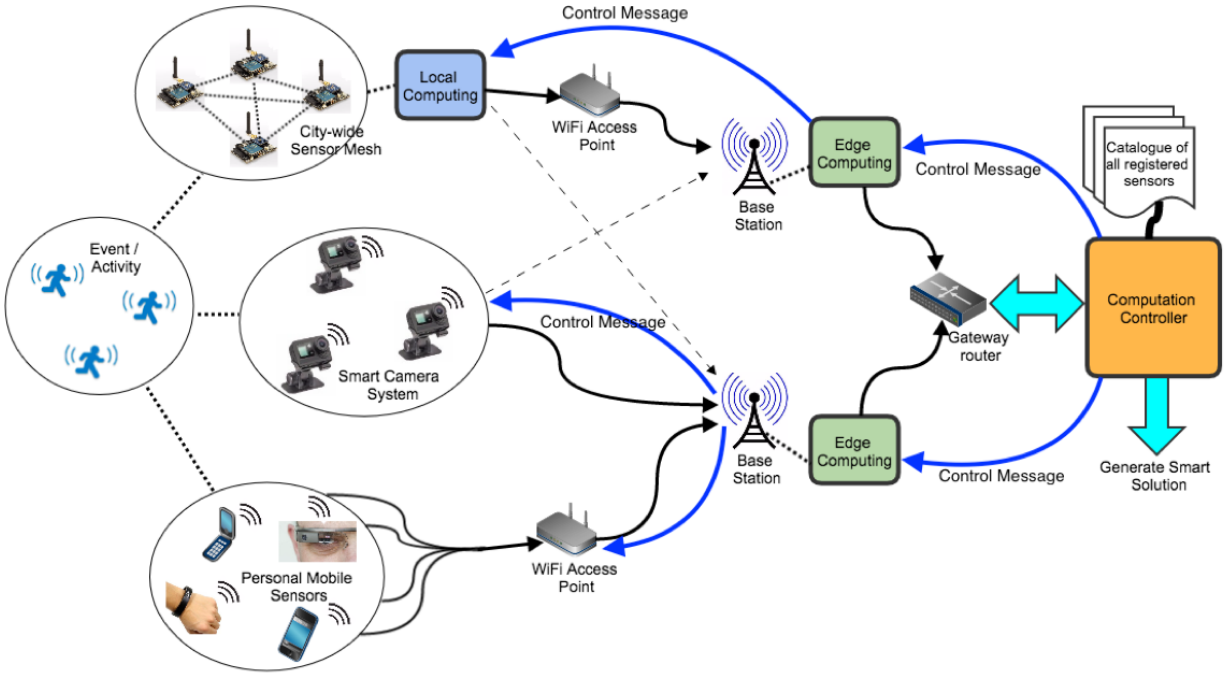


Figure 1.1: Multi-scale Computations in the urban IoT.

In order to mitigate the problems of inefficient on-board computations in terms of task completion time or energy consumption, multi-scale computation can be used where computing can be done in a remote server which is much powerful than the on-board computer. However, for real-time computations in the urban IoT, the end-to-end delay constraints may not be met if the computations done at the end could server which is at the other side of the backbone network. To overcome this problem, edge computing paradigm is used where computing devices which is more powerful than the on-board computer but much less powerful than the end cloud server is placed. In most cases, the edge servers are sufficiently capable

of processing the computations for intelligent and autonomous systems in almost real-time. The edge computing servers can also facilitate to forward the computed results to the end cloud for further non-real-time analysis, thus saving sending the same raw data stream again to the end cloud - hence improve the network utilization. As shown in Figure 1.1, the edge computing server can coordinate with the base station or the wireless access point to facilitate real-time controls for transmission of the data over wireless networks based on various constraints, e.g., latency, energy, temperature or other constraints. However, even with employing edge computing, the impact due to variability in the wireless environment and heterogeneity of the coexisting applications needs to be addressed and resolved.

1.3 Dissertation Contributions & Overview

This dissertation aims at solving the aforementioned challenges using the adaptive communications involving intelligence at both at the sensor and at the edge side. The sensors leverage the structural semantics of the data for content-aware transmission, thus helping a graceful coexistence with heterogeneous technologies and applications. On the edge-side the information awareness can be incorporated through novel a software-defined architecture for helping real-time computations. It also contributes to the adaptive communication using multipath and multi-hop communications, thus optimally using the available network resources. We use the notion of probe to predict the channel quality of the unused paths and making intelligent decision to optimize the Quality of Service (QoS) of the application while ensuring minimal impact on coexisting applications. We demonstrated the solutions on real-world use case scenarios - some in simulation and some on systems implementations.

The following is the overall organization and research contributions of the thesis:

- Chapter 2 presents a literature survey on related works on the main problem and the

sub problems on communications and computations in the urban IoT.

- Chapter 3 address the problems of interference management for an urban IoT application. It considers a real-world video streaming as an application and leverages the encoding property of the data to create a Markov decision process based on the probability of success of the delivery of the video stream and the coexisting data stream. Our key contribution is that making the coexisting application aware about the content (priority) and context (position in the encoding window) can improve the gain of throughput of the coexisting application while maintaining a dynamic Quality of Computation (QoC) requirement. It also shows how to avoid starvation of the non-priority user in case of multiple priority users operating.

- Chapter 4 introduces a novel approach for a software defined Network Function Virtualization (NFV) to support real-time edge computations. It develops a low cost solution for using built-in kernel feature on Linux called extended Berkeley Packet Filter (eBPF) to develop a dynamic data filtering model that can save communication and system resource usage and also minimizes latency of simultaneous computations on the same data. We show the performance of the system with real-world implementation.

- Chapter 5 investigates using alternate communication resources to efficiently support the computation while balancing the performance cost. It defines a threshold based network selection among LTE and WiFi networks through edge-assisted feedback to the sensors. The solution while not being optimal gives intuitions of optimizing the performance using multipath solutions.

- Chapter 6 designs and develop a novel probe-based predictive network selection model for real-time video streaming applications. Our key contribution is effectively and efficiently predicting the alternate unexplored paths for future QoS while minimally impacting the coexisting users. The solution first generates datasets with realistic traffic traces and learns the prediction model which is implemented on a online version

for real-time path selection. The model shows significance performance improvement compared to single channel naive or bitrate adaptive single channel performance.

- Chapter 7 describes building a fully open-source integrated UAV-network simulator to facilitate various communication scenarios for mission oriented applications over UAV, which are difficult to implement exhaustively in real-world scenarios. As a contribution to the open source community the simulator is released for open use for research.

- Chapter 8 investigates the feasibility of infrastructure assistance for autonomous UAV systems. It primarily studies the two mission oriented applications including remote navigation assistance and computation task offloading over WiFi and LTE infrastructures over different communication ranges and network loads. It provides results showing the necessity of adaptive communications for mission-oriented applications.

- Chapter 9 demonstrates building a robust prototype of real-world airborne systems supporting distributed mission oriented computations. In proposes a predictive models including learning joint behavior of UAV control and communication and predict the performance of the mission. It also proposes multi-hop communication and distributed multipath enabled computation and communication architecture over UAV systems.

- Chapter 10 concludes the dissertation with summarizing the overall approach of proposed adaptive communications that can support various applications in urban IoT and discusses about potential future research directions.

Chapter 2

Related Work

In context of urban IoT and its challenges, several related research works have been going on to solve different challenges, e.g., some works aimed at optimizing the performance accuracy while some care about timeliness; some may more care about minimizing the resource usage while some prioritizes fairness in coexistence. In this section we primarily survey related works on different computing architectures in urban IoT, also various types of mission-critical applications that we are interested in and lastly various communication paradigms which are the main aspects of this thesis.

2.1 Computation Architecture in IoT

Traditionally, the multi-scale computing architecture in the urban IoT is hierarchical, the bottom level at the device is least powerful and resource constrained whereas the top level at the cloud server is with very high computational power and with abundance of the resources. In the following subsections, we describe different approaches and intelligence incorporated at different level of the multi-scale computing architecture to facilitate various applications.

2.1.1 Cloud Computing

Traditional cloud computing, i.e. processing and storing the IoT data in a remote cloud server has been useful for various IoT applications that do not require real-time response. For example, in [49], the authors describes the challenges in converging the IoT and the cloud computing. The paper [155] proposes a decentralized distributed mini cloud storage for storing IoT data for flexibility in storage and access. The paper [75] develops a framework for efficient processing and managing healthcare data from wearable devices. In another work [56], a cloud computing architecture is proposed for IoT big data acquisition, management and mining is proposed. The paper [178] shows a smart home IoT appliance and its integration with cloud through web services. In another work [99], a cloud for IoT data from vehicular technology is built, primarily for two use cases - smart parking and vehicular data mining. In all the aforementioned works, the applications are not highly time-critical thus can afford to have the computation delays from the cloud servers. However, for efficient computation offloading to the cloud, the end-to-end communication need to be good.

2.1.2 Edge Computing

In order to minimize the computation delay, computation is proposed at one hop wireless distance away from the sensors - i.e., at the wireless edge [176]. The IoT usecases that are mission-critical but also computation heavy can get benefit from this edge computation paradigm. Especially, if there is a real-time control loop back to the sensor from the computation server is involved, edge computing [52] is necessary. The edge server can be connected to a WiFi access point (Cloudlet) or to a cellular base station (Mobile edge). The paper [71] makes a comparison for long range Mobile edge computing to the computing at the cloudlets at shorter range. In [129] an edge-computing based deep learning framework is proposed for serving the IoT applications. A light weight virtualization framework at the edge is proposed

in [153] to facilitate fast and small computations for IoT applications. In [180] a blockchain based distributed control is proposed for edge computing. Although the edge computing is promising for computations in IoT for real-time service, it still depends on the delay over wireless networks which has high variability, specially in urban environment.

2.1.3 On-board Computing

On-board computation is the simplest computation architecture where the sensor side is responsible for all the computations. However, due to the computation challenges on the sensor side described earlier, the on-board computation may lack in quality of computation or in latency and resource usage. To mitigate the problem, splitting the computation between sensor and edge is proposed so that, sensor can execute partial data with lower load [125].

2.2 Applications in Urban IoT

In this section, we describe the time sensitive and/or mission-critical IoT applications and different adaptive protocols in the literature that facilitates the applications and services.

2.2.1 Multi-sensor Data Acquisition

Several works on multi-sensor based data fusion including [151] and [208] describes the challenges of fusion of heterogeneous sensor data. A multi-sensor data fusion for robotic control is described in [139]. Clearly, these data fusion for real-time control systems like robotics need very low latency and high precision and also synchronization among the data sources. In paper [167] a decentralized multi-sensor based real-time tracking and surveillance system is proposed. All these aforementioned applications add complexity in computation in

terms of synchronization, load and latency constraints. Hence, effective methodologies with efficient algorithms are needed to support the multi-sensor data fusion for various smart services in the urban IoT.

2.2.2 Real-time Video & Vision Applications

Real-time video streaming is one of the major application in the urban IoT that requires high bandwidth and low latency. In paper [21] an edge-computing based video analytics framework is demonstrated. A traffic control application using real-time video streaming is proposed in [194] by number plate matching to check illegal vehicles. In paper [191] a real-time large scale video analytics framework for smart city is proposed. However, in the dynamic environment of urban IoT, the quality of video streaming gets compromised in terms of either data losses, hence corruption in the video frames or delay in play out. To mitigate the quality degradation, adaptive algorithms, e.g., adaptive bit rate (ABR) based algorithms [15] or adaptive framerate [211] solutions are proposed. [59] designs and develops a self-optimizing wireless video application that can adapt in real-time based on context and energy constraints. An urban traffic surveillance video stream using fog computing is proposed in the paper [62]. Apart from the video streaming, with the advent of faster real-time object detection algorithms, e.g. Yolo [168] new applications including surveillance, public safety, traffic management etc. are developing in the urban IoT that needs stringent latency and computation requirements. To cope with the resource constrained computing devices, light wight framework for object detection, e.g. Mobilenetv2 [171] and Yolo-lite [105] are developed.

2.2.3 Autonomous UAV based Applications

Unmanned Aerial Vehicles (UAV) are envisioned to perform articulated tasks in complex mission scenarios. Apart from traditional task of carrying and delivering objects from places to places, UAVs can be equipped with many sensors including video sensors, location sensors, mini radar, sensors to monitor environment and many others to perform newly developed tasks for the urban IoT. To this aim, they continuously sense the surrounding environment and process the generated data with sophisticated algorithms. However, commercial UAVs inherently operate under limited on-board resources. As a consequence, intense computation tasks often incur a delay sufficiently large to impair the ability of the UAVs to autonomously control their navigation and operations. However, there are various applications based on autonomous operation of UAV developed in the realm of IoT. In paper [150] a intelligent transport system based on UAVs is proposed for smart city. A UAV-based disaster management framework is proposed in [76]. Another important use of UAV as envisioned in recent times, is to increase the network coverage and connectivity as a communication relay or a flying base station as proposed in papers [140, 19]. However, in this thesis we focus on real-time sensing and control applications over UAVs in urban IoT.

2.3 Communications in Urban IoT

The main focus of the thesis is the communications to support smart and intelligent applications in the urban IoT. In the previous section, we discussed about the challenges in communication in the urban IoT. Here, we discuss about different communication paradigms incorporated in cities for handling the traffic demand and also coping with the variability in the network conditions.

2.3.1 Infrastructure-assisted Communications

In the infrastructure-assisted communication, the popular communication modes are WiFi and Long Term Evolution (LTE). WiFi while operates on the unlicensed frequency bands in 2.4 GHz and 5 GHz bands, based on family of IEEE 802.11 standards [154]. The increased demand of bandwidth over wireless networks gave rise to proposing carrier aggregation or channel bonding in WiFi standards [199]. LTE traditionally operates in the licensed frequency spectrum and is regulated by 3GPP [86]. Usually WiFi works in the shorter range within around 100m range, while LTE works for long range communication over few miles. WiFi follows distributed coordination function for resource allocation whereas LTE is a cellular technology that centrally controls the resource allocation through as base station. Recent proposition of LTE suggests to use LTE also in unlicensed frequency band [216] as well as a licensed assisted mode. Using these infrastructure, new communication paradigms are also evolved, e.g. communication of vehicle to everything (V2X) over LTE [23, 173] and WiFi [66], smart grid automation over LTE [79], Line-of-sight detection [219] and indoor localization [185] using over WiFi infrastructure etc. The development of these upcoming technologies and their enhancements encourage to implement and improve adaptive algorithms for communications over the urban infrastructure.

2.3.2 Ad-hoc and Device-to-device Communications

Due to the large volume of data traffic over the wireless infrastructure along with mobility and other variations, ad-hoc communications between devices is used to improve the data transmission. Over WiFi, the device-to-device (D2D) communications is done as WiFi-direct [26] and over LTE, it is done as LTE-direct [67] or as LTE underlay communications [72]. However, most of the adhoc communication happens in the unlicensed band requiring an interference management with coexisting users using the same frequency bands.

Wireless and Device-to-Device (D2D) communications are specially beneficial in high speed mobility scenario like vehicular or UAV to UAV communications.

2.3.3 SDN-based Core Networks

To support the computation at the cloud and edge, flow control algorithms that is supported by Software-defined networking (SDN) can be incorporated. In [193], the opportunities and challenges of deploying IoT applications over SDN is proposed. An end-to-end orchestration of IoT services with Network Function Virtualization (NFV) is proposed in [200, 161]. In paper [82] an SDN based improved security is proposed for IoT applications. A deep learning based traffic load prediction and adaptive communication for IoT applications is presented in [186]. The primary advantage of SDN is that it can be programmed as per the computation requirement dynamically, thus can provide optimal trade off for the usage of the communication resources. However, there are several challenges to develop SDN based solutions, especially for time-sensitive and mission critical applications which we address in this thesis.

Chapter 3

Content-Aware Cognitive Interference Control for Urban IoT Systems

3.1 Introduction

The Internet of Things (IoT) is a recent communication paradigm that enables the interconnection and interoperation of everyday life objects, equipped with transceivers for digital communication and suitable protocol stacks [29]. An emerging trend proposes the application of the IoT paradigm to the urban environment [213, 47], that is, the Urban IoT, which integrates city-wide sensing, communication and processing resources to improve the efficiency of public services.

We contend that the cognitive network paradigm [109, 143] can play an important role in the design of networking technologies capable of supporting communications and computations in the complex Urban IoT scenario. One of the main challenges arises from the heterogeneity of the technologies supporting communications at different geographical scales, which often coexist on, and compete for, a shared channel resource. Importantly, in some application

scenarios, communications supporting Urban IoT systems utilize the same frequency bands used by traditional wireless services. There is, then, a strong need for technological solutions facilitating the coexistence of heterogeneous data streams, especially for those applications where algorithms processing data streams in real-time may impose stringent and short term Quality of Service (QoS) requirements to the network.

Here we propose a new framework, where cognitive techniques are used to control the interference generated by mutually interfering data streams, some of which may transport information critical to accomplish real-time IoT computational tasks. The main observation behind the framework proposed in this study is that different portions of data streams supporting IoT services may have different relevance to the global computational goal. The framework we propose realizes a cognitive *content-oriented* channel access control, where transmission of concurrent data streams is shaped to create optimal interference patterns to critical IoT data streams. We refer to the former and latter as cognitive and "protected" data streams. However, we remark that in a general formulation the IoT streams can also implement cognitive behaviors.

In most traditional cognitive network frameworks, the notion of cognition was used to identify and use for transmission of empty time/frequency slots, the so called white space [90]. The objective, then, was to make the communications of the different classes of users orthogonal, avoiding interference as much as possible. In modern network coexistence scenarios, the portions of spectrum shared by multiple technologies may be occupied by many users transmitting heavy data streams, such as multimedia information. In this context, a rigid classification of users typical of traditional cognitive network frameworks jointly with a white space-based approach would result in the starvation of non-prioritized users.

The framework we propose seeks practical methodologies to control the interference generated by the cognitive wireless nodes to concurrent data streams in a semantic fashion. The transmission pattern of the cognitive users is defined based on the state of the concurrent

data streams, defined to describe the relevance of each segment of data being transmitted to the algorithms consuming the data. However, while the binary idle-busy state necessary to control transmission in white space-based approaches can be directly inferred by the cognitive nodes, the semantic information driving transmission decisions in the proposed framework necessitates a sophisticated architecture entailing the collaboration of devices and infrastructure-level agents.

A key component of the proposed architecture is edge computing [53], where computation resources (edge processors) are placed within a one-hop low-latency topology to provide real-time and reliable services within wireless islands. In our proposed architecture, the edge processors support transmission control by analyzing the incoming data streams to extract useful semantic information. The wireless nodes, the cellular resource manager and the edge processor collaborate to build a map of the content state in the time/frequency slots occupied by the protected IoT users.

To illustrate the general principle behind the proposed architecture, we consider a case-study scenario where Device-to-Device (D2D) communications coexist with Long-Term Evolution (LTE) cellular communications in the same bandwidth [73]. The chosen application is remote monitoring, where video streams acquired by local sensors are analyzed by the edge processor to detect and classify objects of interest, for instance to identify treats and support autonomous transportation networks.

We assume that the IoT data streams transport encoded video over LTE, and that a local, cognitive, D2D link uses for transmission the same channel resource. Interference from the D2D link can severely impair the performance of algorithms processing the video streams in real time. In order to minimize its impact on the decoding and analysis processes performed at the edge processor, the cognitive transmitter exploits the semantic structure of the temporal encoding of the video streams.

The proposed scheme requires the extension of the LTE protocol stack to include the notion of content state, and to implement messaging supporting the content-aware transmission strategy of the cognitive terminals. We conducted our investigations by implementing these modifications within the NS-3 simulation environment. By means of detailed network and data processing simulations, we demonstrate that the proposed content-oriented paradigm can significantly increase the throughput achieved by the D2D transmitter for a given maximum degradation of the processing algorithm output compared to a uniform interference strategy.

3.2 Analytical Model on Content-Based Interference Management

Device-to-Device (D2D) communications have been envisioned as an effective solution to support local information exchange without generating additional traffic to the network infrastructure. Herein, we consider a network scenario where a D2D link operates on a Long-Term Evolution (LTE) uplink channel [51, 73, 221, 83] as an underlay.

The main challenge is the control of mutual interference, especially due to the logical separation between the two networks. Recent contributions proposed mode selection and power control based interference control mechanisms, where the objective is to maintain the Signal-to-Interference-plus-Noise-Ratio (SINR) at the LTE receiver above a predefined threshold [221, 111, 217]. In this study, we propose a novel content-based interference strategy, which targets interference on specific packets within the information stream. We contend that this construction is especially suited to the Urban IoT, where data streams from sensors are compressed, and the relevance of the transmitted information might change over time.

We consider a scenario where the LTE link is transporting compressed video, and the D2D

transmitter is bound by a constraint on the minimum Peak Signal-to-Noise-Ratio (PSNR) measured at the LTE receiver.

The dynamics of the network are modeled as a Markov process [48], which tracks the transmission process of the LTE link and the transmission decisions of the D2D transmitter. The optimization problem is formulated as a Markov Decision Process (MDP) problem, and solved by means of a specialized Linear Program (LP). The output of the optimization problem is a randomized past-independent policy [116], that is, the transmission action is solely a function of the current state of the system.

The obtained optimal transmission strategy and achieved performance demonstrate that the D2D link can significantly improve its throughput if interference is targeted to specific packets within the video stream. More specifically, the D2D transmitter should reduce interference generated to the LTE link when reference frames are transmitted. Results obtained by applying the optimal policy to real-world video show that the PSNR significantly improves with respect to that of the reference transmission policy.

3.3 Network and Application Scenario

Fig. 3.1 depicts the network considered in this study, where LTE and D2D links coexist on the same channel resource and mutually interfere. The LTE mobile terminal is uploading a video to the network infrastructure, whereas the D2D device is assumed to be transmitting best-effort traffic to a neighboring mobile device. In the following, we first describe the video compression techniques that are widely used for storage and transmission, and, then, the physical layer model adopted in the analysis and optimization framework.

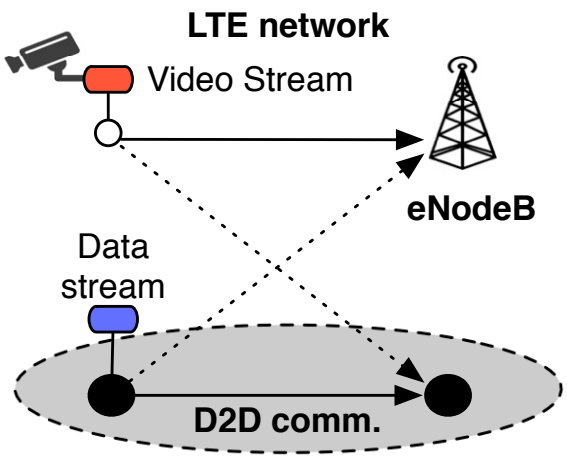


Figure 3.1: Network configuration considered in the study. The LTE and D2D links coexist on the same channel resource and mutually interfere.

3.3.1 Video Streaming:

A video is a sequence of images produced at a given rate. Unlike images, which only have a spatial component, a video has a temporal component as well. Therefore, compression is performed both along the spatial and temporal dimensions.

Spatial compression: The individual pictures are broken into "macroblocks" which are transformed by Discrete Cosine Transform (DCT) from space domain to frequency domain. Basic MPEG-4 uses 8×8 DCT whereas H.264/MPEG-4 Advanced Video Coding (AVC) standard uses a 4×4 DCT-like integer transform. The transformed data, then, are quantized and encoded using entropy coding.

Temporal compression: Temporal compression exploits the significant similarities that may interest pictures within the video stream captured at close time instant. This gives the opportunity to encode the video in a smaller number of key (reference) frames and a larger number of differential frames following the reference frame, which encode differences with respect to neighboring reference frames. Thus, in order to decode differential frames, the decoder also uses the associated reference frame(s).

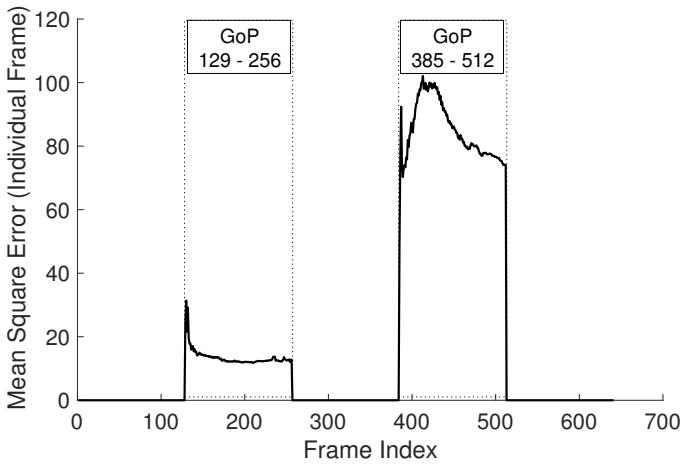


Figure 3.2: MSE as a function of the frame index. The I-frames with indexes 129 and 385 are heavily damaged and the error propagates through the entire GOP (128 frames).

When an encoded frame is damaged, that is, packets transporting fragments of it are lost or received with excessive delay, due to spatial compression, the decoding process results in the multifold corruption of the output image content and resolution. The spatial propagation of errors may create artifacts that are detected as objects, or impair the ability of the algorithm to detect existing objects. Additionally, due to temporal compression, if a reference frame is damaged, the effect propagates through the entire Group of Pictures (GoP) defined as a reference frame and a set of dependent differential frames. Conversely, if a differential frame is damaged, the effect remains localized to the frame[1], as some information regarding motion vectors may be lost, but key features in following frames are decodable. Herein, we consider H.264/AVC encoding, where GoP starts with intra-coded frame (I-frame) followed by a number of inter coded differential frames (D-frame), that is, Predicted frames (P-frames), Bi-directional predicted frames (B-frames) and can be of constant size or variable size.

To illustrate error propagation in a GoP, we emulated the effect on a real-world video from a surveillance camera. The video is composed of 641 frames and is highly compressible, with fixed GOP size equal to 128 frames. Each frame is composed of 640x360 pixels. In Fig. 3.2, we show the impact of the loss of the I-frames with indexes 129 and 385 on the per frame

[1]Although some dependencies may exist across differential frames

Mean Squared Error (MSE) of the video. The error propagation effect is perceivable. Thus, D2D transmission during a reference frame transmission carries a larger amount of "effective interference" to the computing algorithm compared to the same transmission during differential frames. Note that artifacts caused by damage to reference frames may persist during the entire GoP and mislead object tracking algorithms to categorize them as objects.

Based on this observation, we define content classes which include a flag discriminating between reference and differential frames. Thus, we build the notion of relevance of packets within periods of the data stream based on the semantic structure of the data stream itself. In a more general definition, the content class within periods could be defined by the presence of objects of interest (*e.g.*, pedestrians), and the state would need to be extracted by the edge processor.

3.3.2 Physical Layer

Analogously to most prior literature on D2D underlaying LTE networks, we abstract the physical layer using a per-packet decoding threshold. Let's denote the LTE and D2D transmitter and receiver by the subscript ℓ and d, respectively. Due to mutual interference, the SINR at the LTE and D2D receiver are

$$\mathrm{SINR}_\ell = \frac{P_\ell c_{\ell\ell}}{\sigma_\ell^2 + P_d c_{d\ell}}, \quad \mathrm{SINR}_d = \frac{P_d c_{dd}}{\sigma_d^2 + P_\ell c_{\ell d}}, \tag{3.1}$$

respectively, where P_ℓ and P_d is the average received power at the LTE and D2D receivers (including path loss), c_{xy} is the fast fading gain from node $x \in \{\ell, d\}$ to receiver $x \in \{\ell, d\}$ and σ_{x}^2 is the noise variance at receiver x.

We assume that the coefficients c_{xy} have Rayleigh probability density function, that is,

$$F_{c_{xy}}(c) = P(c_{xy} < c) = 1 - e^{-c}, \quad c \geq 0, \tag{3.2}$$

where the power of the fading process is 1. Here, we consider a slotted time model, where the fading coefficient is assumed constant within each slot, and fading coefficients in different slots are independent and identically distributed (*i.i.d.*) random variables. In order to provide an intuitive and insightful model, we assume that each slot corresponds to the transmission of one video frame. Then, the probability that $SINR_x$ is above the decoding threshold γ and the packet is successfully decoded is

$$q_x = \frac{e^{-\frac{\gamma\sigma_x^2}{P_x}}}{1 + \frac{\gamma P_y}{P_x}}, \tag{3.3}$$

with $x \neq y$. Or we can say that the error probability is

$$\rho_x = 1 - q_x = 1 - \frac{e^{-\frac{\gamma\sigma_x^2}{P_x}}}{1 + \frac{\gamma P_y}{P_x}}, \tag{3.4}$$

Note that LTE may fragment frames into multiple packets. In this case, our assumption corresponds to invariant fading for the duration of the transmission of the burst of packets associated with a single video frame.

3.4 Stochastic Model

We develop a stochastic model to study the performance of the wireless nodes in the co-existence scenario described earlier, and optimize the transmission strategy of the D2D transmitter. Our objective is to measure the PSNR of the video link and the throughput of the D2D link as a function of the transmission strategy of the D2D transmitter. Clearly, one

of the key aspects to capture is the error propagation effect due to the differential encoding used in video compression. In this analytical framework, we assume that the loss of a reference frame causes the loss of all the frames in the GoP. A numerical study of this effect in real-world video streams is provided in Section 5.5. The definition of models capturing bidirectional dependencies are left to future studies.

We build the stochastic model to track the number of frames transmitted and delivered within each GoP. We logically divide the temporal evolution of the process into renewal periods where the first state corresponds to the transmission of an I frame, and the subsequent states in each period correspond to the transmission of differential frames. Note that the D2D link is a best effort data stream, where we only track the delivery of individual packets. We, then, define the Markov process $\mathbf{S}=(S(0), S(1), S(2), \ldots)$, where $S(t)\in\mathcal{S}$ is the state in slot t and $\mathcal{S}$ is a finite state space. The state of the system is described by the vector $(I_{\text{rx}}, N_{\text{tx}}, N_{\text{rx}})$. The variable $I_{\text{rx}}\in\{0,1\}$ is equal to 1 if the I frame associated with the current GoP was correctly decoded at the video stream receiver, and is 0 otherwise. N_{tx} and N_{rx} are the number of differential frames transmitted and received within the GoP, respectively, with $0\leq N_{\text{rx}}<N_{\text{tx}}\leq N$, where N is the maximum GoP size. The, possibly variable, size of the GOP is modeled by defining the distribution β, where $\beta(i)$ is the probability that the GOP terminates after the transmission of i differential frames, with $\beta(N)=1$. Note that fixed size GOP encoding can be modeled by setting $\beta(i)=0$ for $i=1,\ldots,N-1$. We also define the action variable $U(t)\in\{0,1\}$, where 0 and 1 correspond to idleness and transmission of the cognitive D2D transmission, respectively. Note that different transmission power can be incorporated as decision variables. Herein, we fix the transmission power of the two terminals, and limit the decision of the cognitive terminal to a binary variable. Then, we define $\rho_x(u)$ as the failure probability of the packet sent by link $x\in\{\ell, d\}$ conditioned on the decision variable u. Note that $\rho_d(0)$ is trivially equal to 0, whereas $\rho_\ell(0)=1-e^{-\frac{\gamma\sigma_\ell^2}{P_\ell}}$. When the links mutually interfere, the failure probabilities can be calculated from equation (4) as

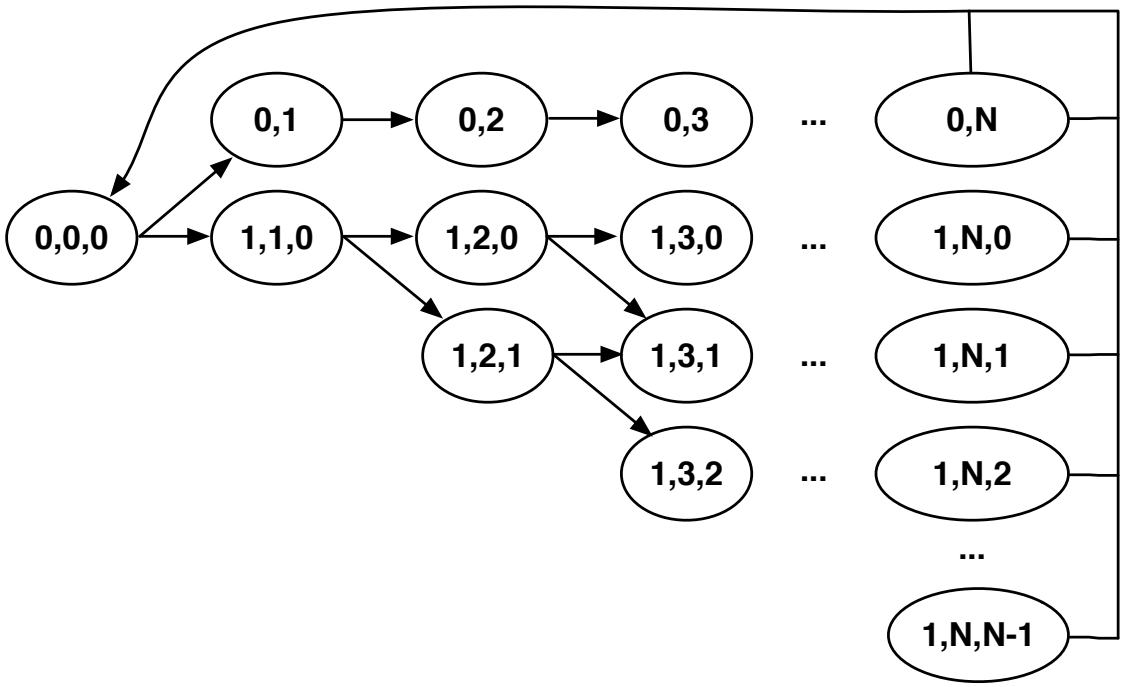


Figure 3.3: Representation of Markov chain for a case with fixed GOP length.

$$\rho_d(1)=1-\frac{e^{-\frac{\gamma\sigma_d^2}{P_d}}}{1+\frac{\gamma P_\ell}{P_d}}, \quad \rho_\ell(1)=1-\frac{e^{-\frac{\gamma\sigma_\ell^2}{P_\ell}}}{1+\frac{\gamma P_d}{P_\ell}}. \tag{3.5}$$

Fig. 3.3 depicts the state space and allowed transitions.

The state $(0,0,0)$ corresponds to the transmission of the I-frame. Conditioned on action u, the process then moves either to $(1,1,0)$ or $(0,1,0)$ with probability $1-\rho_\ell(u)$ and $\rho_\ell(u)$, respectively. The former and latter state indicates successful and failed decoding of the I-frame. All the states in the first row have N_{rx} value 0 as per our assumption of losing all frames in GOP when reference frame is lost.

From state $(i_{\text{rx}}, n_{\text{tx}}, n_{\text{rx}})$, $0<n_{\text{tx}}<N$, and conditioned on action u, the Markov chain moves

to the states

$$(0,0,0) \text{ with probability } \beta(n_{\text{tx}}), \tag{3.6}$$

$$(i_{\text{rx}}, n_{\text{tx}}{+}1, n_{\text{rx}}{+}1) \text{ w.p } (1{-}\beta(n_{\text{tx}}))(1{-}\rho_\ell(u)), \tag{3.7}$$

$$(i_{\text{rx}}, n_{\text{tx}}{+}1, n_{\text{rx}}) \text{ w.p } (1{-}\beta(n_{\text{tx}}))\rho_\ell(u), \tag{3.8}$$

where $\beta(n_{\text{tx}})$ is the probability that the GoP terminates at the transmission of the n_{tx} differential frame, and another I-frame is sent. From state $(i_{\text{rx}}, N, n_{\text{rx}})$, the process moves to the states $(0,0,0)$ with probability 1. since the GoP terminates. Note that in the transition probabilities listed above, the selection mechanism for the control variable is left unspecified. In the next section, we will formulate an optimization problem whose output is the control action distribution defined on the state space of the network.

3.5 Performance Metrics and Optimization Problem

3.5.1 PSNR Analysis

Our first goal is to derive an evaluation metric, *i.e.*, the PSNR, which is commonly used to measure video quality. We, then, compute the Mean Square Error (MSE) of the video sequence averaged over all frames, and from this value we obtain the PSNR as

$$\text{PSNR} = 10 \log 10(K_{\text{bps}}/\text{MSE}), \tag{3.9}$$

where K_{bps} is 2^W and W is the number of bits per pixel.

The technique we use for the evaluation of the PSNR is an extension of that proposed in [183, 31]. We briefly sketch it and then discuss how it applies to our evaluations. In [183],

the MSE of the received video is seen as the sum of multiple uncorrelated distortion values. Hence, we write $MSE{=}\mathcal{D}_e{+}\mathcal{D}_u$, where $\mathcal{D}_e$ is the distortion introduced by the encoder and $\mathcal{D}_u$ is the distortion term caused at the decoder by residual errors, *i.e.*, packet error. Herein, we only focus on $\mathcal{D}_u$ and assume $\mathcal{D}_e$ as constant, as it does not depend on packet and frame loss. According to the framework proposed in [183], which has been validated via extensive simulations, pixel errors caused by not delivered frames can be approximated by Gaussian distributions. Moreover, it is also assumed that further manipulations of the received signal performed by the codec are linear and time-invariant, and thus can be represented through the frequency responses of some filters. If this filtering is applied iteratively, based on the central limit theorem, one can expect that the impulse response of the filter also becomes Gaussian after a sufficiently large number of iterations [7]. Thus, we model the term $\mathcal{D}_u$ as a Gaussian variable with zero mean and variance σ_u^2, where the latter linearly depends on the frame error rate. Following the considerations in [31], this assumption leads to $\mathcal{D}_u{=}C\sigma_u{=}Cp_{\text{err}}\sigma_e$, where C and σ_e are constants and function of the video and encoding/packetization implementation, and p_{err} is the frame error rate.

3.5.2 Performance Metrics

The PSNR, then, is monotonically increasing in the frame delivery rate, which we use to evaluate the performance of the video. Within the adopted markovian framework, the frame delivery rate is measured as

$$D_{LTE}(\mu) = \lim_{T\to\infty} \frac{1}{T} E\left[\sum_{t=0}^{\infty} \mathbb{1}(\Omega(t))\right], \tag{3.10}$$

where μ is the control strategy and $\Omega(t)$ is the event corresponding to a frame delivery in slot t. $D_{LTE}(\mu)$ can be rewritten as

$$D_{LTE}(\mu) = \sum_{s\in\mathcal{S},u\in\{0,1\}} \pi_\mu(s,u)\omega(s,u), \tag{3.11}$$

where the function $\omega(s,u)$ counts the frames transmitted in the current GoP, that is,

$$\omega(s,u)=\beta(n_{\text{tx}}) * i_{\text{rx}} * (n_{\text{rx}}+i_{\text{rx}}+1-\rho_\ell(u)) \tag{3.12}$$

in state $s=(i_{\text{rx}}, n_{\text{tx}}+1, n_{\text{rx}}+1)$. The term $\beta(n_{\text{tx}})$ accounts for the fact that n_{rx} is the number of frames received in the buffer only if the frame terminates. The multiplicative term i_{rx} forces to zero the number of frames delivered if the I frame failed.

The normalized throughput of the D2D link, measured in successfully delivered packets per slots, is defined as the long-term average

$$T_{D2D}(\mu) = \lim_{T\to\infty} \frac{1}{T} E\left[\sum_{t=0}^{\infty} \mathbb{1}(\Phi(t))\right], \tag{3.13}$$

where μ is the control strategy and $\Phi(t)$ is the event corresponding to the delivery of a packet by the D2D link. The above measure can be rewritten as

$$T_{D2D}(\mu) = \sum_{s\in\mathcal{S},u\in\{0,1\}} \pi_\mu(s,u)\phi(s,u), \tag{3.14}$$

where $\pi_\mu(s,u)$ is the joint state-action steady state distribution

$$\pi_\mu(s,u) = \lim_{t\to\infty} P_\mu(S(t)=s, U(t)=u), \tag{3.15}$$

and $\phi(s,u)=1-\rho_d(u)$.

3.5.3 Optimization Problem

Based on the stochastic model and performance metrics defined earlier, we can now define the optimization problem aimed at maximizing the D2D link throughput under constraints on the PSNR. The optimization problem

$$\mu^* = \arg\max_{\mu} T_{D2D}(\mu) \;\; \text{s.t.} \;\; D_{LTE}(\mu) \geq \delta, \tag{3.16}$$

admits at least one optimal policy μ in the class of randomized past independent policies [116]. We focus on this class of policies, and define the policy $\mu(u,s)=P(U(t)=u|S(t)=s)$. Then, the problem (4.2) is mapped to the Linear Program (LP) [116, 128]

$$\begin{aligned}
\mathbf{z}^* = \arg\min_{\mathbf{z}} \quad & 1 - \sum_{s,\omega} \phi(s,u)\, z(s,u) \qquad (3.17)\\
s.t. \quad & \sum_{s,u} \omega(s,u)\, z(s,u) \geq \delta \\
& \sum_{s,u} z(s,u) p(s'|s,u) = \sum_{\omega \in \mathcal{A}} z(s',u), \\
& \sum_{s,u} z(s,u) = 1, \\
& z(s,u) \geq 0, \;\; \forall s, \omega,
\end{aligned}$$

where for the sake of readability, we denote $\sum_{s\in\mathcal{S}} \sum_{u\in\{0,1\}}$ with $\sum_{s,u}$ and $\mathbf{z}$ as the vector $\{z(s,u)\}_{\forall s,u}$, and $p(s'|s,u)$ is the transition probability to s' conditioned on the state s and action u. In these formulation, the optimization variable $z(s,u)$ is the the joint steady-state probability of the state-action pair (s,u), that is, $z(s,u) = \Pr(S(t)=s, U(t)=u)$. The optimal policy μ^*, then, is

$$\mu^*(s,u) = \frac{z(s,u)}{\sum_{u'\in\mathcal{A}} z(s,u')} \;\; \forall s,u. \tag{3.18}$$

The high level abstraction of the proposed model is used to derive guidelines for the definition of adaptive transmission protocols in heterogeneous application and network scenarios. Clearly, the application of such policies in a real-world scenario necessitates advanced capabilities of the terminals and network devices, and should account for some important networking aspects.

Firstly, we observe that in practical networks, the interference strategy needs to be defined at the LTE Medium Access Control (MAC) level, where video frames are fragmented into several LTE packets. These packets are, then, transmitted over the channel. Heuristics based on the optimal policy can be directly applied to this case, where the transmission probability is a function of whether the packets being transmitted belong to an I-frame or D-frame, as well as whether or not the I-frame in the GOP was significantly damaged. We describe the implementation of these strategies in the LTE protocol stack in later sections.

3.6 Numerical Results of the Analytical Model

We now present the simulation setup and numerical results assessing the performance of the proposed interference scheme.

3.6.1 Simulation Setup

We compare the optimal policy to a baseline case where the transmission probability of the D2D transmitter is equal to p_{tx} in all the states. In this case, for fixed GOP size, the delivery

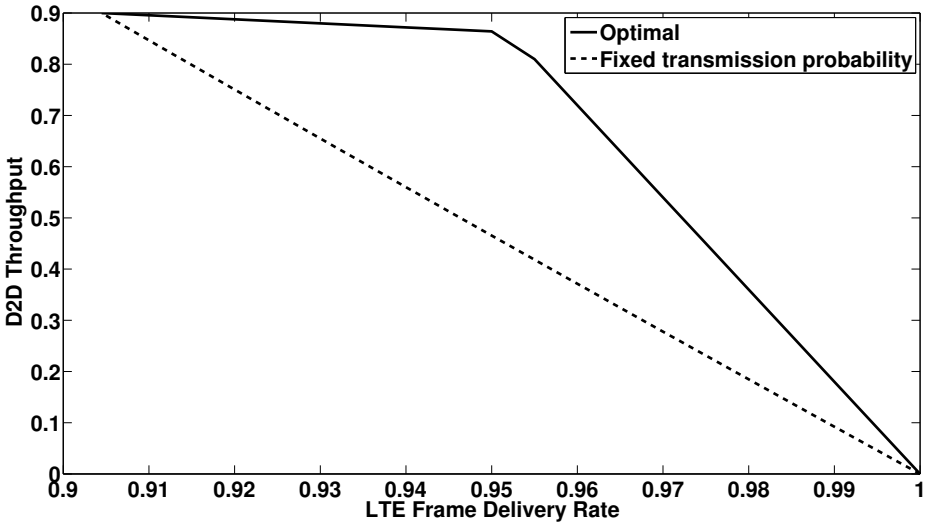


Figure 3.4: Throughput of the D2D link as a function of the LTE frame delivery rate for the optimal (solid line) and the fixed transmission probability (dashed line) policy.

rate of frames is

$$\begin{aligned} D_{LTE}(p_{\text{tx}}) &= \left[\sum_{i=0}^{N} \frac{N!}{i!(N-i)!}(1-\rho_\ell(p_{\text{tx}}))^i \rho_\ell(p_{\text{tx}})^{N-i}\right] \\ &= \frac{(1-\rho_\ell(p_{\text{tx}}))(N(1-\rho_\ell(p_{\text{tx}}))+1)}{N+1}, \end{aligned} \tag{3.19}$$

where

$$\rho_\ell(p_{\text{tx}}) = \rho_\ell(1)p_{\text{tx}} + \rho_\ell(0)(1-p_{\text{tx}}). \tag{3.20}$$

The throughput of the D2D link is trivially equal to

$$T_{D2D}(p_{\text{tx}}) = p_{\text{tx}}(1-\rho_d(1)). \tag{3.21}$$

For this analytical model we characterize the LTE and D2D links by their error probabilities. The error probabilities are set to $\rho_\ell(0)$=0.01, and $\rho_\ell(1)$=$\rho_d(1)$=0.1. We simulate the linear programming solution of the optimal policy and the baseline solution using MATLAB. For

our simulation, we assume a fixed GOP size equal to 24 frames, including the I-Frame. We also take a real world video of same GOP size to show the propagation effect of the I-frame loss in entire GOP.

3.6.2 Results

Fig. 3.4 shows the throughput of the D2D link as a function of the LTE frame delivery rate for the optimal (solid line) and the fixed transmission probability (dashed line) policy. In the fixed transmission probability case, the throughput of the D2D link decreases linearly as the delivery rate of the LTE frames increases. The optimal policy significantly increases the throughput of the D2D link until the constraint on the LTE frame delivery becomes too tight, and the D2D transmitter is forced idle.

Fig. 3.5 depicts the transmission strategy of the D2D link as a function of the constraint on the minimum LTE frame delivery rate. The dashed and dotted lines correspond to transmission probabilities when an I-Frame and a D-frame are transmitted. In the latter case, only frames where the I-frame in the same GOP was delivered are considered. In fact, the optimal policy trivially prescribes transmission with probability 1 in states corresponding to D-frames whose I-frame was lost. When the minimum LTE delivery rate is equal to or larger than the maximum (approximately 0.98), the D2D transmitter is idle in the state corresponding to I-frame transmission. As the constraint decreases, the D2D transmitter increases the transmission probability in states corresponding to D-frames transmission, whereas the policy prescribes idleness in states corresponding to I-Frame transmission. As the transmission probability in D-frames reaches 1, the D2D transmitter begin to increase the transmission probability in the state corresponding to I-frame transmission. The optimal policy, then, avoids interference to the I-frames to minimize damage to the video until the minimum frame delivery rate is large enough to tolerate the cascade effect triggered by

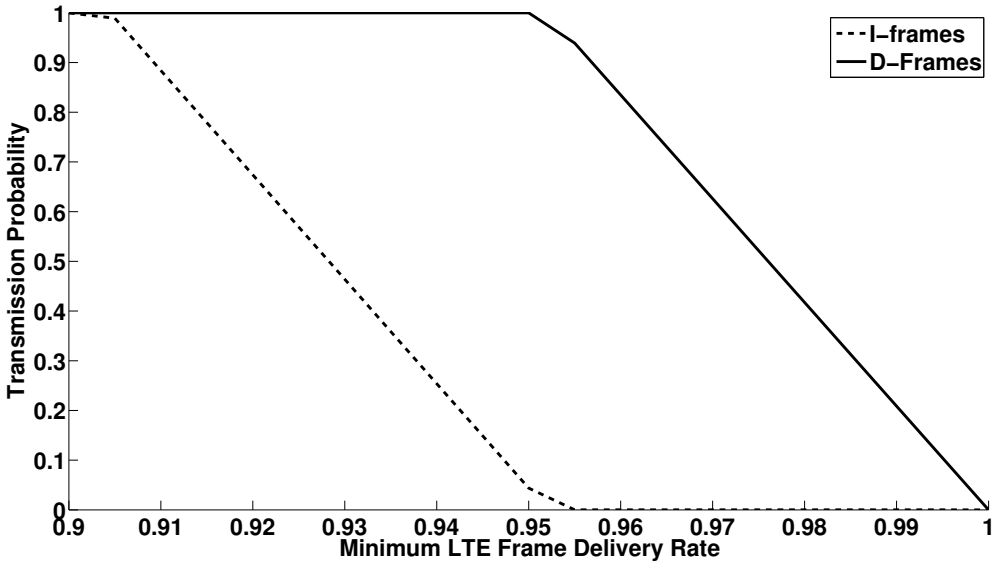


Figure 3.5: Transmission strategy of the D2D link as a function of the constraint on the minimum LTE frame delivery rate. The dashed and dotted lines correspond to transmission probabilities when an I-Frame and a D-frame are transmitted.

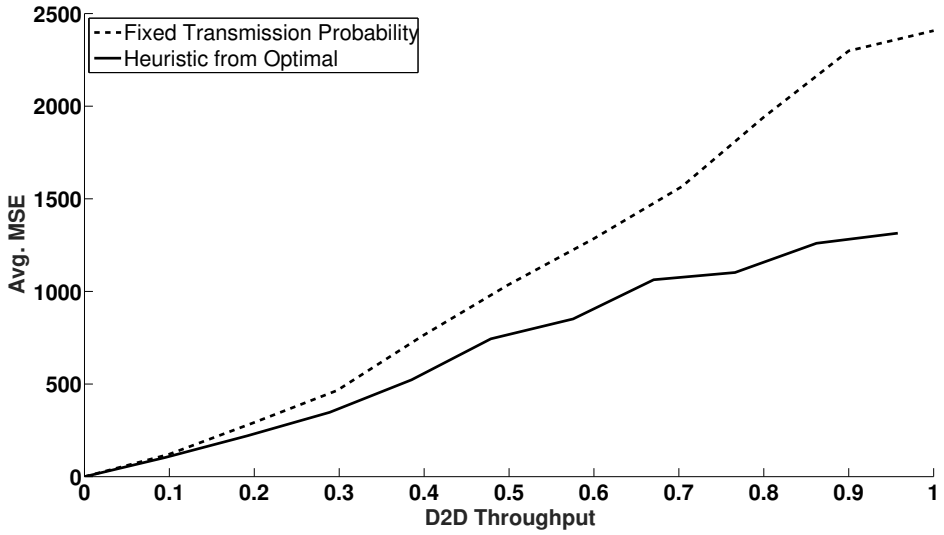


Figure 3.6: Average MSE as a function of the throughput achieved by the D2D link.

I-frame loss events.

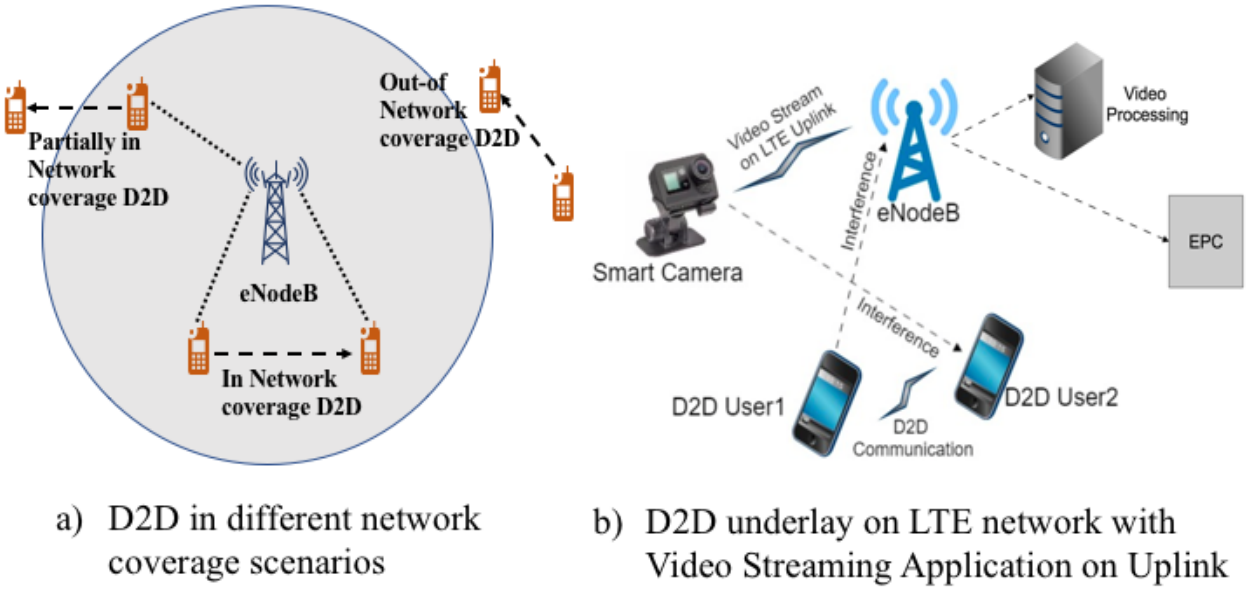
Finally, in Fig. 3.6 we show the MSE as a function of the D2D throughput for the video in the previous picture. The shown results are obtained using the fixed transmission probability policy, and a heuristic extracted from the optimal policy, where the transmission probability is set to 0 when an I-frame is transmitted, and is constant otherwise. The points in the

plot correspond to different values of the transmission probabilities. Note that this policy is simpler to implement in real-world networks, as it requires much less coordination compared to a policy defined on the full state space of the Markov process modeling the dynamics of the network. In the plot, the failure rates were set to $\rho_\ell(0)=0$, $\rho_\ell(1)$ and $\rho_d(1)=0$ to increase the impact of D2D transmission on the LTE link and improve convergence rate of computationally demanding simulations. It can be observed that the heuristic policy significantly reduces the MSE, especially in the high throughput region. We remark that the parameters chosen in these simulations result in strong interference to the LTE receiver when the D2D transmitter is active.

3.7 Evolution of D2D and LTE coexistence in the Urban IoT

Due to spectrum scarcity, wireless communication and network technologies are being designed to coexist on the same portion of spectrum. For instance, recent LTE-based cellular networks are proposed to operate in unlicensed 5 GHz band [216] to offload traffic. Since this band, as per IEEE 802.11 standards, is also used by Wi-Fi, an interference control problem arises. The development and increasing popularity of Device-to-Device (D2D) communications further exacerbates the problem of interference, especially in dense urban deployments. Originally, D2D communications were only designed for short range links without infrastructural support. More recently, WiFi direct has been proposed for direct D2D communications between WiFi devices without coordination from a local Access Point (AP).

D2D communications over cellular networks were originally proposed as a way to increase network connectivity [132]. Recent studies use this technology to build high data rate, short-range, links between devices and increase spectral efficiency [50, 73, 163]. Different



a) D2D in different network coverage scenarios

b) D2D underlay on LTE network with Video Streaming Application on Uplink

Figure 3.7: D2D scenarios and ProSe architecture in 3GPP LTE

mechanisms have been proposed for this class of D2D communications. Recent work studied D2D communications as an in-band overlay of cellular networks, where a dedicated portion of the spectrum is assigned to D2D communications. However, this family of solutions may not be efficient in terms of bandwidth utilization and is not scalable [83, 130]. Alternatively, D2D communications can be deployed in-band, where D2D transmissions underlay the spectrum used by the LTE cellular infrastructure. However, in-band D2D underlay may result in a reduced ability of the network to control interference.

The proliferation of local services (*e.g.*, social networking) led to the inclusion of D2D communications in the recent release 12 of 3GPP [13] as Proximity Services (ProSe). In the LTE standard, the D2D link is defined as a "sidelink" in order to differentiate it from uplink and downlink. The major two features defined in the standard are D2D discovery and D2D data communication. D2D discovery is signaled by the Physical Sidelink Discovery Channel (PSDCH), whose assigned resources are specified by the eNodeB in the Downlink Control Information (DCI). Herein, we do not address the D2D discovery mechanism and, instead, focus on D2D data communication assuming the D2D devices are connected and ready to

communicate. Specifically, we consider D2D communications "in-network coverage" or "partially in network coverage" as shown in Fig. 3.7(a), where the D2D transmitter interferes with uplink LTE transmissions.

Resource allocation can be performed in "scheduled" or "autonomous" mode. In scheduled mode, the eNodeB allocates the resources for D2D links to operate using the Physical Sidelink Control Channel (PSCCH) and Physical Sidelink Shared Channel (PSSCH). In case of in network coverage autonomous resource allocation, the D2D device selects portions of a resource pool preconfigured by eNodeB in the System Information Block (SIB) [13].

3.7.1 3GPP LTE and D2D Radio Access Network

Although the methodology proposed in this study can be applied to a wide range of scenarios, we consider a network setting where data streams are transmitted over Frequency Division Duplex LTE (FDD-LTE) networks for multi-scale processing, and a D2D terminal transfers data using D2D underlay of LTE in the same frequency band as shown in Fig. 3.7(b). We assume network-assisted D2D communications [12], where the LTE operator assists in establishing the D2D connection through authentication and authorization over E-UTRAN. Once the connection is established, data communication is performed via direct radio rather than through the E-UTRAN infrastructure, which can be used for occasional control and signaling messages. The framework we propose is positioned within the in network coverage autonomous resource allocation configuration, and we assume the cognitive D2D devices use control information from the LTE eNodeB to synchronize with the LTE frame structure.

Data from the LTE User Equipments (UE) are streamed uplink via the Serving Gateway (SGW) and Packet Data Network Gateway (PGW) in the Evolved Packet Core (EPC) network towards a remote host. Herein, we are primarily interested in uplink transmission by E-UTRAN. When the RRC (Radio Resource Control) layer of UE has a Packet Data Unit

(PDU) to transmit on uplink, it passes the PDU to the underneath PDCP (Packet Data Convergence Protocol) layer which forwards it to the RLC (Radio Link Control) sublayer. The RLC reports its buffer status to the underlying Medium Access Control (MAC) layer, which transmits a control message to the eNodeB MAC scheduler. The scheduler allocates Resource Blocks (RBs) for data transmission based on the channel quality measured in terms of Signal-to-Interference-plus-Noise-Ratio (SINR) which varies with received signal strength, noise and interference. As per the standard, for LTE uplink, the SINR is measured using a control signal, e.g., the SRS (Sounding Reference Signal), or a data signal, e.g., PUSCH (Physical Uplink Shared Channel), at the eNodeB MAC layer. The eNodeB layer calculates the CQI and reports it to the scheduler. The scheduler selects a suitable MCS and notifies the UE MAC via DCI [3]. The UE MAC then transmits the PDU(s) based on the data rate supported by the assigned MCS. When the D2D link is active in the LTE uplink band, the interference it causes degrades the achievable channel rate of the UEs. If the interference is high, the channel may not be able to support even the lowest MCS, in which case the LTE receiver fails to decode the PDU.

Prior contributions addressing interference control in D2D communications underlaying LTE proposed scheduling and interference control strategies that aim at maintaining the SINR at the LTE receiver above a certain threshold [165, 210, 201]. However, these techniques require instantaneous channel knowledge for every transmission slot as well as physical layer parameters, and they may not harvest all the available bandwidth in this complex coexistence scenario. Herein, we assume that the D2D transmitter autonomously makes transmission decisions within the assigned spectrum without any prior information on the channel. D2D transmissions interacts with the rate selection mechanism described earlier, and may cause the delayed delivery or the loss of PDUs.

3.7.2 Interference Control Problem

In order to make our description general, we refer to the data transmission slots (subframes in the LTE architecture) as "periods", where the i–th period contains the data generated by the transmitter in the time interval τ_i=$[t_{i-1}, t_i)$, i=$1, 2, 3, \ldots$, with t_0=0. We map each period to a "state", which we denote with S_i. The state describes the content generated by the UEs in the corresponding section to the processing algorithm. We define the finite state space $\mathcal{S}$=$\{1, \ldots, K\}$, with $S_i \in \mathcal{S}$, i=$1, 2, 3, \ldots$, and the sequence $\mathbf{S}_i^N$=$(S_{i-N+1}, \ldots, S_i)$. Further, we define the set of packets generated in τ_i as $\mathbf{p}_i$.

Transmission of the D2D transmitter is controlled by a policy μ, which is defined over the same synchronized time periods as the content state of the interfered data streams. The action takes the form of a transmission probability function of the state within each period. The action in the time period i, then, is a_i=$\mu(\mathbf{S}_i, \mathbf{a}_{i-1}) \in \mathcal{A}$, where $\mathcal{A}$ is a finite action set and $\mathbf{a}_i^N$=$(a_{i-N+1}, \ldots, a_i)$.

The packets generated by the LTE UEs suffer an interference pattern that is a function of the policy. Herein, we assume that the packet loss pattern in τ_i is a (probabilistic) function of the action a_i. For the sake of generality, we say that due to interference, the network transforms the packet sequence $\mathbf{p}_i$ into the sequence $\hat{\mathbf{p}}_i$, where some packets may be missing due to decoding failure, excessive delay, *etc.*

The output of the algorithm depends on the content state and received packet sequences. The performance of the algorithm consuming the incoming data stream is measured by the function $\phi(\hat{\mathbf{p}}, \mathbf{S})$, where $\hat{\mathbf{p}}$=$(\hat{\mathbf{p}}_1, \hat{\mathbf{p}}_2, \ldots)$. In this study, we accomplish the following objectives:

Architecture: Develop an architecture supporting content-aware transmission policies capable of: *a)* promptly identifying the content state of incoming data streams; and *b)* notifying D2D transmitters to drive transmission actions.

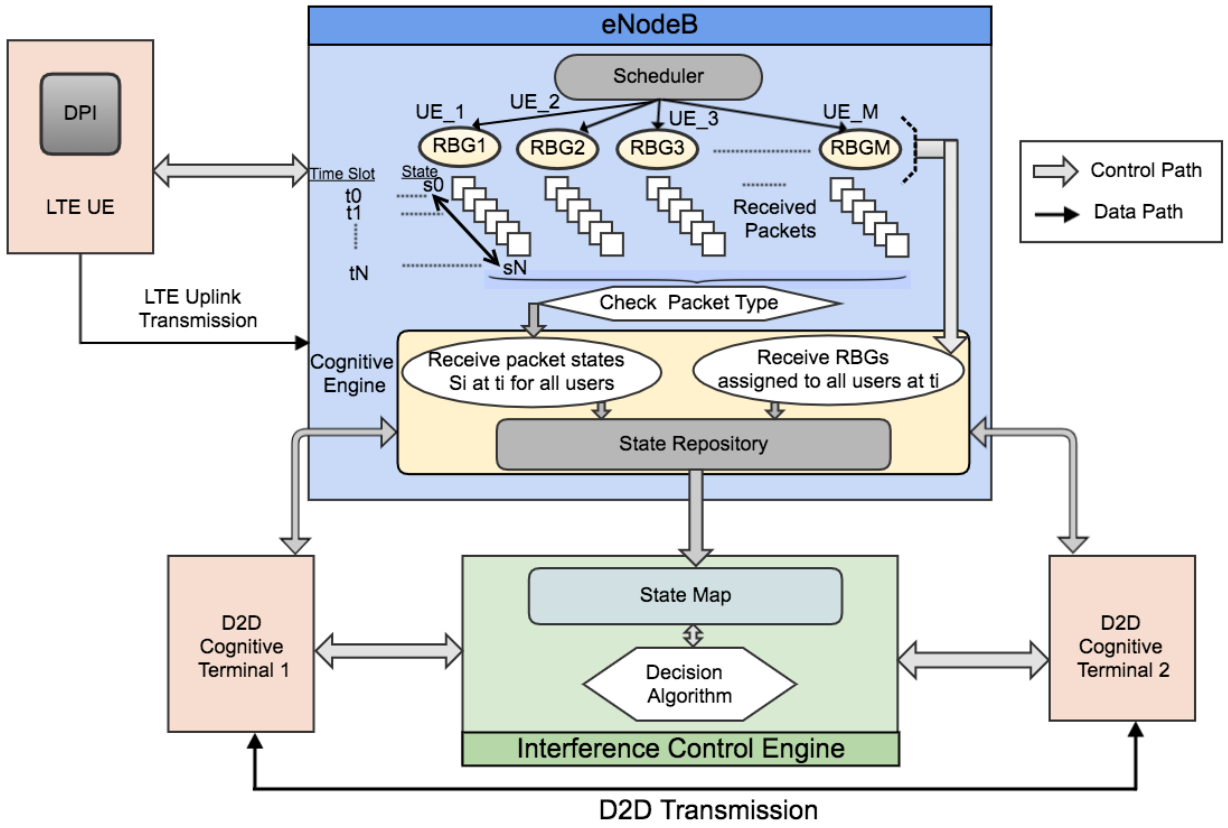


Figure 3.8: System Architecture for D2D underlay with LTE Uplink Transmission.

Transmission Policy: Devise content-aware transmission policies μ controlling the transmission pattern of the D2D users whose objective is to preserve the quality of the output of algorithms processing the data streams, that is, maintain $\phi(\hat{\mathbf{p}}, \mathbf{S})$ above a predefined threshold. In this study, we use Q-Learning, a popular form of Temporal-Difference learning to identify the optimal transmission, and thus interference, policy. As clarified within the context of the specific scenario considered in this study, due to the mapping on periods, rather than packets as in most prior work in the cognitive networking domain, the action space can include articulated actions corresponding to sub-policies applied within each period.

3.8 System Architecture

First, we describe the architecture supporting the content-aware interference policies operated by the D2D link. The system architecture defines the components of the cognitive network and the control and data path for the LTE and D2D messages. Along with those characterizing the LTE stack, the components include a Deep Packet Inspection (DPI) module which interacts with the MAC layer of the LTE UEs, and a cognitive engine which contains a state repository collecting the required information for supporting the transmission decision algorithm. In the following, we describe in detail these components, which are summarized in Fig. 3.8 together with the message flows as described in Fig. 3.9.

Deep Packet Inspection: The UEs implement a DPI module in the MAC layer to determine the packet type at the source. This functionality allows the early identification of packet classes generated by applications and data encoders. The class might already be a sufficient descriptor of the content state S_i. Later in the study, we connect this model to a specific scenario, where the type of frame emitted by a video encoder determines the relevance of packets to accomplish a computational objective, which takes the form of object detection and classification. In this case, the "content state" is constituted of the packet type based on the specific encoded data frame it belongs to, and the position of the packet in a frame or group of frames. The UE transmits a control message via the Uplink Control Information (UCI) on Physical Uplink Control Channel (PUCCH) with the content state to the eNodeB which first schedules a Resource Block Group (RBG) to the user and then assigns a MCS based on the CQI as shown in the Fig. 3.9. The eNodeB sends the content type of the allocated users and the scheduling map to the "cognitive engine" for processing. Note that, the UEs do not report content state for every individual packets, but only when a state change is detected in order to reduce control message overhead for the DPI messages.

Cognitive Engine: The cognitive engine resides in the eNodeB, which collects the content

states and resource allocation of the UEs for each transmission period and creates a "State" repository. The eNodeB also collects information regarding the spectrum assigned to the D2D transmitter via UCI and builds an optimized map of the content state in each subframe and LTE channel.

The state repository in the cognitive engine maintains the state of the last N packets for each UE to determine if the next incoming packet belongs to the same continuing period or is the beginning of a new period. Based on the state queue, the cognitive terminals can employ instantaneous or history dependent interference control strategies. The cognitive engine is notified by the D2D terminals regarding the resource pool it is using, and hence, determines the m overlapping UEs out of total M UEs ($m \in M$) as scheduled in the subframe.

Decision Algorithm: In the context of LTE FDD, one LTE radio frame of 10 ms duration is divided into 10 subframes of 1 ms duration each and each subframe contains 2 slots of 0.5 ms. The radio resources are allocated in terms of Physical resource block (PRB) [3] of 12 subcarriers such that each slot of 0.5 ms contains a number of PRBs. The decision algorithm controls the transmission of the D2D link in each subframe based on the interference control strategy. Thus, the action set $\mathcal{A}$ corresponds to transmission probabilities ρ_i within individual subframes. For simplicity, we assume that D2D transmissions occupy the whole assigned spectrum of the resource pool. However, the transmission policy can be extended to include the dynamic selection of one or a subset of channels based on the state map.

Fig. 3.10 shows an example of packet flow of one protected and multiple cognitive data streams in the cognitive system architecture. The proposed content-oriented policies shape the transmission process of the cognitive nodes to create interference patterns compatible with the interfered data stream and its processing.

We define two classes of heuristic transmission strategies: Past Independent and Past Dependent protocols.

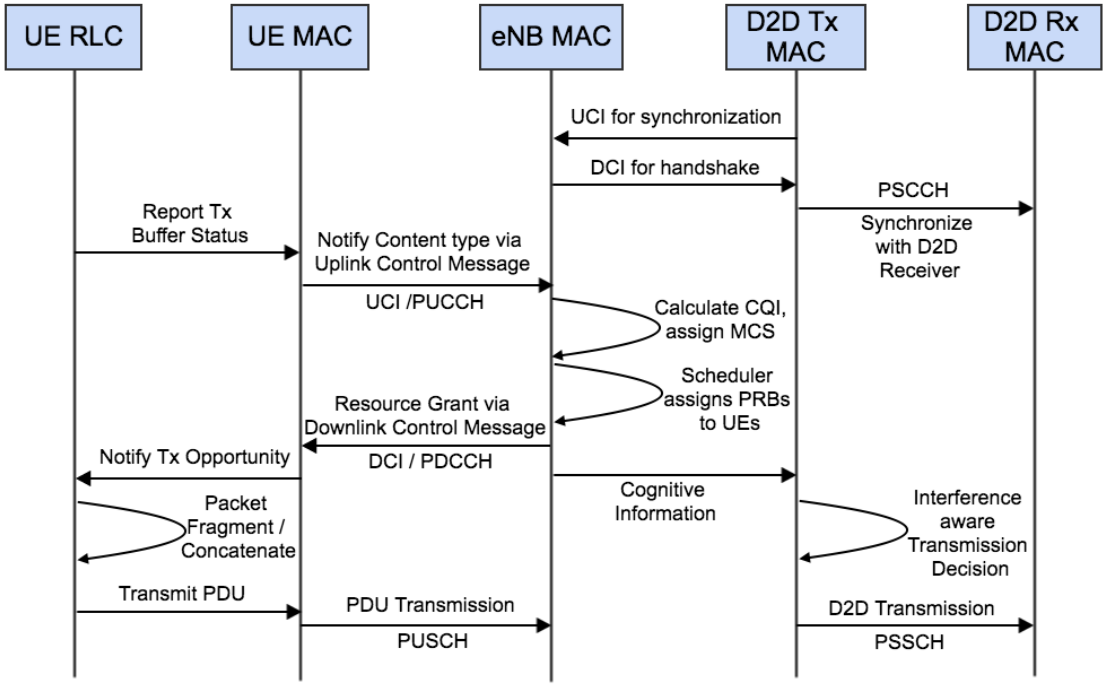


Figure 3.9: Message flow for LTE Uplink Transmission and coexisting D2D Transmission

Past Independent Protocol: In this first class of policies, we define control actions solely on the content state in each individual period S_i.

Past Dependent Protocol: In this second class of policies, the decision in period i are a function of the state sequence $\mathbf{S}_i^N = (S_{i-N+1}, \ldots, S_i)$.

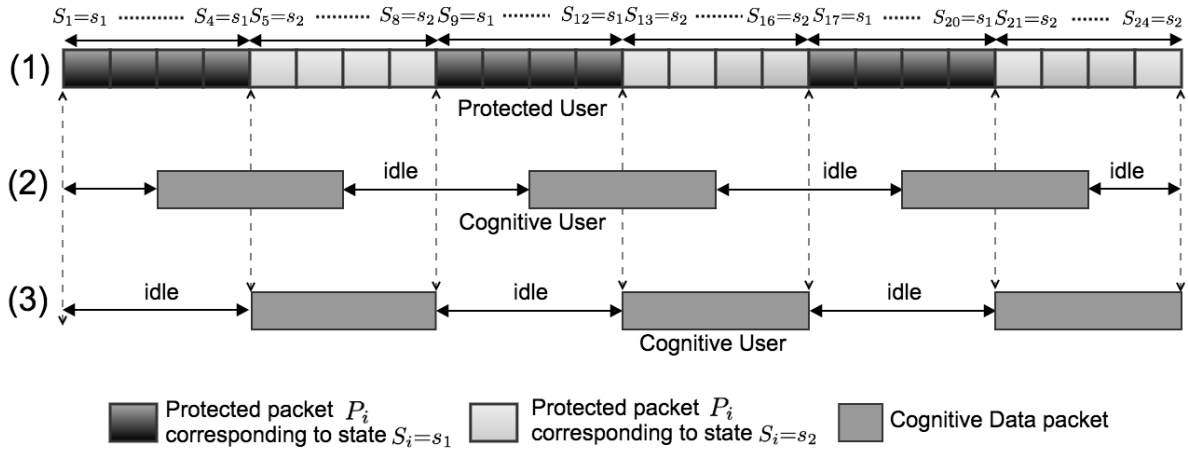


Figure 3.10: Cognitive controlled transmission (1) Protected packet flow, (2) Content-agnostic cognitive packet flow, (3) Content-aware cognitive packet flow

These policies are compared with those obtained by means of Dynamic Programming techniques maximizing a global reward function evaluated at the edge processor.

3.9 Cognitive Interference Control in coexisting LTE and D2D underlay

In order to illustrate its performance, we present an implementation of the proposed framework in a specific scenario, where the video stream from a surveillance camera is processed by the edge processor to identify objects within the individual frames. Referring to the network infrastructure described earlier, the, protected, video streams are transported over LTE, which is interfered by a cognitive D2D link.

In the considered case, the LTE UEs transmitting the video stream sends a preamble message with the control information on the uplink to the eNodeB reporting the frame type (I, B/P). As a GoP contains a series of frames and each frame contains hundreds of packets, it is sufficient to send the frame type at the first packet of each frame instead of in all the packets in that frame. An increased robustness with respect to packet loss can be achieved by repeating the frame type information in more than one packet at the beginning of each frame.

3.9.1 Performance Metrics

We evaluate the performance of the proposed scheme by measuring the accuracy obtained by object detection and recognition algorithms applied to the resulting video stream.

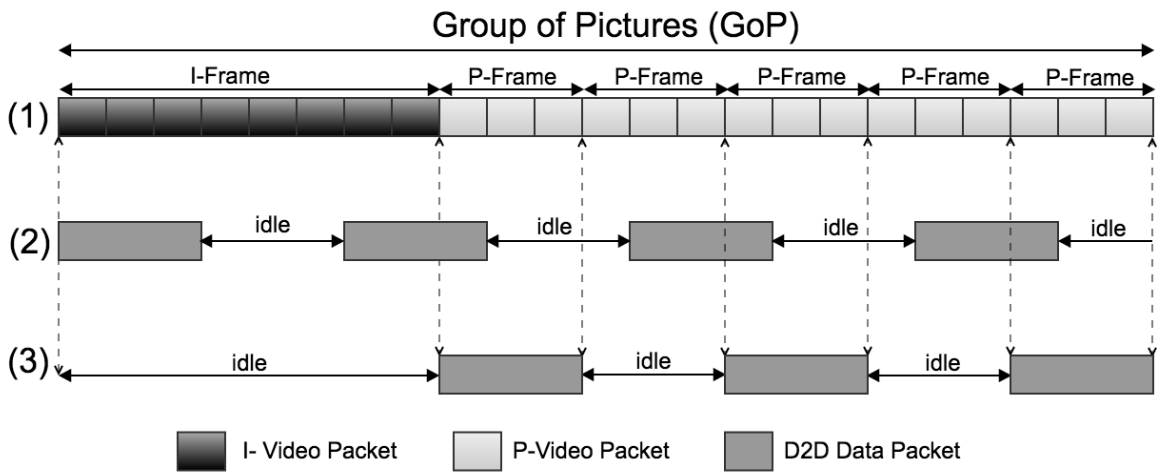


Figure 3.11: Cognitive Interference Strategies: 1) Video packet flow; 2) Fixed D2D Tx probability; and 3) FDTP.

Object Detection

Object detection is one of the primary applications in computer vision. Most object detection algorithms extract features from the image frames and employ a matching or detection algorithm with respect to reference images. In this study, for the purpose of illustration and performance evaluation, we use the Speeded Up Robust Features (SURF) based object detection technique [43] to detect the surface point of objects in each frame of a video with respect to a reference image. So, as interference corrupts the video and affects object detection, we measure the object detection rate as:

$$P_{det} = \frac{N_{rcv}}{N_{ref}} \tag{3.22}$$

where N_{rcv} and N_{ref} are number of correctly detected surface objects in the received and reference video respectively such that $\{N_{rcv}, N_{ref}\} \in N_{orig}$ where N_{orig} is the feature set detected in the original uncorrupted video.

Object Recognition Quality

We also measure the object recognition quality where specific objects are recognized using a classifier trained with a gaussian mixture model [220]. We emphasize that any algorithm in this class is likely to be heavily affected by corruption of the video. In fact, blurred portions of the image, especially when correlated errors occur in the image sequence, can lead the algorithm to detect non-existing objects. Although it is difficult to define a general metric for object recognition, we can measure the object recognition rate, positive prediction rate and the false positive rate with respect to the uncorrupted video based on the "true positive (T^p)", "false negative (F^n)" and "false positive (F^p)" detection as described in [42]:

$$\begin{aligned} Object\ Recognition\ Rate &= \frac{T^p}{T_0^p} \\ Positive\ Prediction\ Rate &= \frac{T^p}{T^p + F^n} \\ False\ Positive\ Rate &= \frac{F^p}{TF} \end{aligned} \tag{3.23}$$

where T_0^p is detected objects in base video and TF is the Total number of Frames.

To measure T^p, F^p and F^n, we compare the object recognition of the corrupted video with the reference video frame by frame. Let N_o be the number of objects recognized in the original video in the frame F. In the corrupted video, the total number of objects recognized is M_o. Then based on how many objects in N_o correspond to the objects in M_o within an error margin equal to δ, we count $m_o \ \epsilon \ M_o$ corresponding objects as true positive for that frame F. The $(M_o - m_o)$ objects which are recognized in the corrupted video but not in the original video are counted as false positive. One point to be noted here that the true negative is 0 as we don't track objects that are not detected. However, the false negatives can be found by the objects which are detected in the original video but not present in the

corrupted video. So, $(N_o - m_o)$ gives number of false negatives.

Efficiency

: We define the efficiency of the control policy as the ratio

$$\text{Efficiency} = \frac{D_{avg}}{1 - T_{d2d}}. \tag{3.24}$$

where D_{avg} is average object detection rate and T_{d2d} is the throughput of the D2D link. Higher efficiency indicates that the LTE UEs achieve high object detection rate, but also provide transmission opportunities to the cognitive D2D terminal.

3.9.2 Interference Control Strategies

First, we seek the optimal transmission strategy of the D2D link by means of Q-Learning, a widely used Temporal Difference Learning Framework. In the considered scenario, we consider a case with one active UE and define states and actions at the temporal scale of GoPs, that is, a reference frame followed by a sequence of differentially encoded frames. Due to the independence between GoPs, the state space is composed of one state only. The simplicity of the state space contrasts with the complexity of actions, which define articulate transmission patterns over the GoP and the associated section of packet stream. In particular, each action is associated with the parameters of two binary Markov chains controlling the transmission pattern of the D2D link in the sections of GoP transporting reference and differential frames. The states of the Markov chain correspond to an idle (state 0) and active (state 1) D2D link, and the global action determines the transmission probability $e = \frac{p_{01}}{p_{10}+p_{01}}$ and average transmission burst size $B = 1/p_{10}$, where p_{ij} is the transition probability between state i and $j \in \{0, 1\}$. A finite set of actions is obtained by

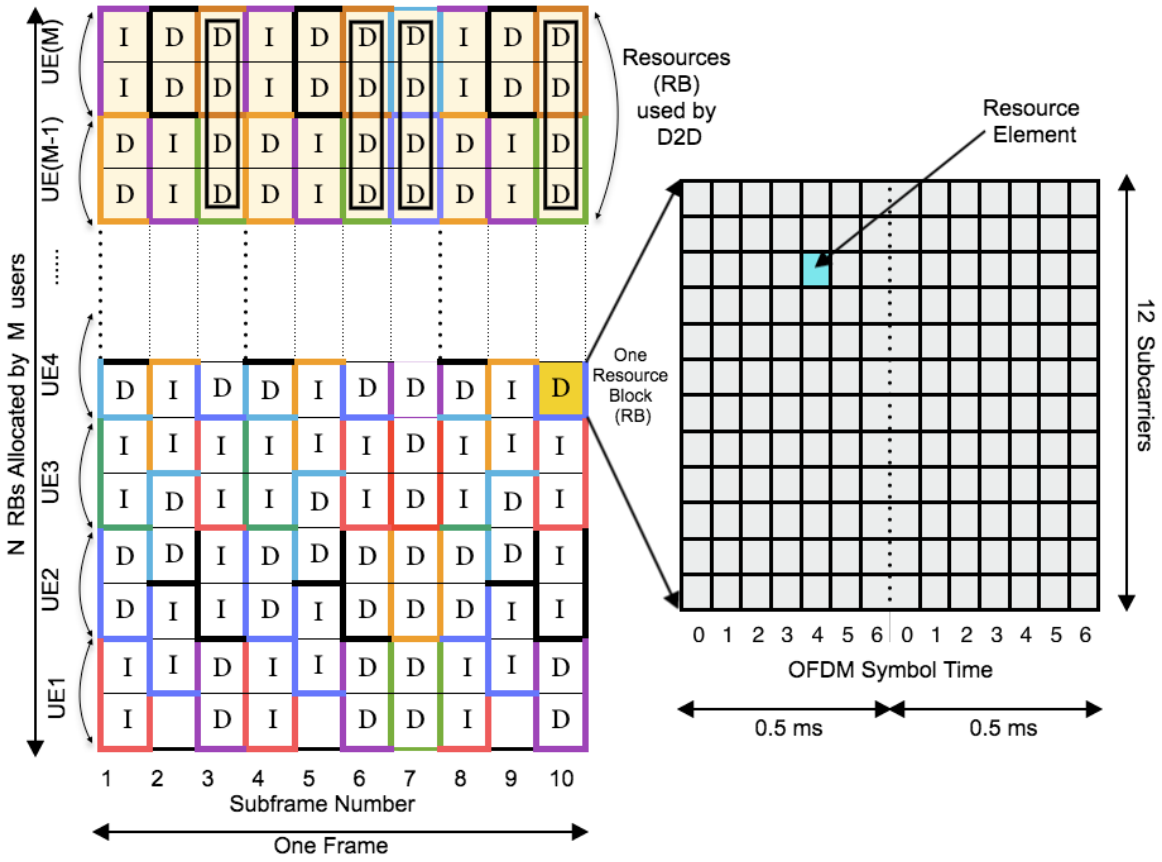


Figure 3.12: State map in a LTE frame with multiple UEs transmitting video frames.

defining a set of e and B parameters for the two chains. A standard Q-Learning algorithm is used to identify the action with the maximum Q-Value computed based on the reward function

$$R(a) = \lambda R_{D2D}(a) + (1 - \lambda) R_{LTE}(a) \tag{3.25}$$

where a is the action in the action space $\mathcal{A}$, $R_{D2D}(a)$ and $R_{LTE}(a)$ are the rewards corresponding to the utility of the D2D and LTE links, respectively, and $\lambda \in [0, 1]$ is a weight parameter. In the considered case. we define $R_{D2D}(a)$ and $R_{LTE}(a)$ as the normalized throughput of the D2D link within the GoP and the average PSNR of the transmitted frames, respectively.

We now instantiate the past independent and dependent policies for the case-study scenario and LTE network environment we described earlier. The policy determines transmission

probability within the LTE frames contained in the time period associated with a data class. Fig. 3.11 provides a graphical representation of the policies presented in this study.

In practical scenarios, several UEs transmitting data stream over LTE can share the spectrum with the D2D. Let us denote the total number of LTE users in the cell as M and total number of resource blocks (RB) in the spectrum as N. So, on average each user is assigned $k = \lfloor N/M \rfloor$ RBs. Suppose, the D2D transmitter uses a resource pool of l RBs (which are usually consecutive). If $(l < k)$, then resources for only one UE (k RBs) will overlap the entire resource pool and the transmission decision of the D2D terminal will be solely based on that specific user's content state. However, when $(l > k)$, then, the D2D transmission decision will be based on more than one overlapping UEs' content type. To be specific, the transmission probability will be dependent on content state of m users where $m = \lceil l/k \rceil$ and $(m \in M)$. The values of k and l are variable; k decreases as number of users M increases and resource pool size l is selected by the D2D. Fig. 3.12 illustrates the resource usage.

The state map, then, is defined to capture the frame type in each subframe and channel, that is, the state in subframe i and RB n is $S_{i,n} \in \{I, D\}$, where I and D correspond to a subframe and RB in which a reference and a differential frame are transmitted, respectively. The state S_i, then, is the vector $S_i = [S_{i,1}, \ldots, S_{i,N}]$.

Frame Dependent Transmission Probability (FDTP): In this past independent protocol, the D2D transmitter selects the transmission probability depending on whether the LTE UEs, are transmitting reference (I) or differential (D) frames in the current subframe. Herein, we simplify the decision policy, and define the transmission probabilities ρ_I and ρ_D, corresponding to subframes where at least one RB is assigned to a UE transmitting an I frame and to subframes where all RB are assigned to UEs transmitting D frames, respectively.

We observe that in a scenario where the spectrum assigned to the D2D link overlaps with multiple UEs, there might be the risk of starvation. In fact, the I frames of the various UEs

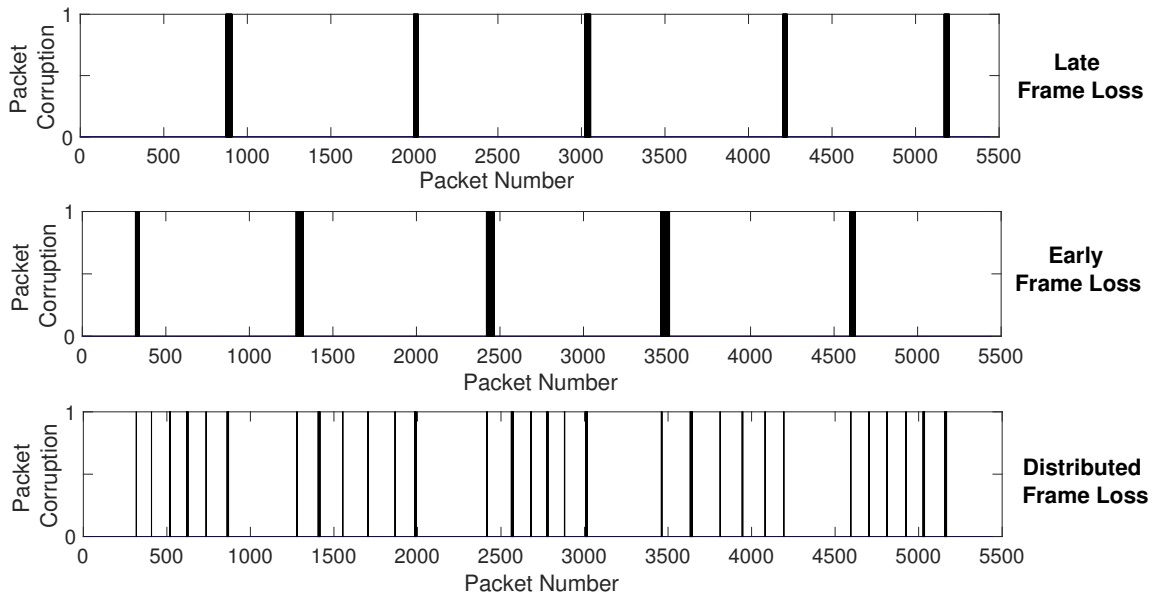


Figure 3.13: History-based interference strategies with different spacing.

can create a thread in which only few subframes exist where all the RBs are assigned to differential frames. To mitigate this issue, more involved policies could be defined to map increasing transmission probability to a decreasing number of RB assigned to UE transmitting I frames.

Frame History Transmission Probability (FHTP): In the FHTP policy, the system exploits the information contained in the state repository, that is, a sequence of past states. With respect to FDTP, we extend the definition of state to include the distance of the corresponding frame within the GoP. The objective is to finely tune the transmission pattern and minimize the impact of interference on the processing algorithms.

First, we analyze the impact of packet loss induced by interference within a frame. Similar to FDTP, we keep subframes transporting reference frames free of interference, and we restrict transmission from the D2D link in subframes associated with PDUs part of differential frames. Unlike FDTP, we concentrate interference bursts in specific differential frames, thus having a smaller number of frames affected by packet loss within each GoP. We experimentally measure the effect of specific patterns with different distribution of frames affected by

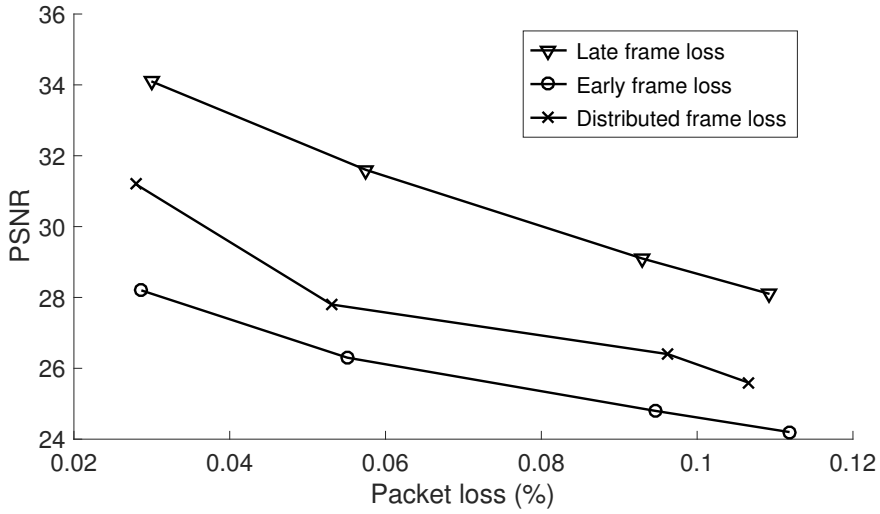


Figure 3.14: PSNR as a function of packet loss for different transmission patterns.

interference bursts within the test video. We select affected frames according to three different patterns; "*Late frame loss*", where we select frames towards the end of GoP to damage, "*Early frame loss*", where we select affected differential frames from the earlier part of the GoP, and "*Distributed frame loss*", where we select affected differential frames at regular intervals within GoP.

Fig. 3.13 shows the packet loss patterns for a number of affected frames equal to 6 in each GoP. We measure the Peak Signal to Noise Ratio (PSNR) to compare the resulting video quality. As each frame contains a variable number of packets, we plot the PSNR against packet loss percentage, where we consider a number of affected frames equal to $6, 10, 15$ and 20. Fig. 3.14 shows that the late frame loss distribution achieves better PSNR compared to early frame loss and distributed frame loss. However, since packet loss in the video corresponds to D2D link transmissions, the distribution also determines its QoS measures. It can be noticed that both the early and late frame loss that concentrate packet loss in a portion of the GoP induce longer idle periods of the cognitive D2D transmitter, which result in larger packet delivery delay. Conversely, distributed frame loss creates transmission opportunities for the D2D link at shorter and regular intervals when differential frames are

being transmitted. Hence, we select the distributed frame loss interference, which provides a good tradeoff between video reconstruction quality and performance of the D2D link.

In order to create this class of interference patterns, we add to the state stored in the repository a descriptor of the damage incurred by specific frames within the GoP. For the sake of exposition, we consider a case in which all the RBs used by the D2D link are assigned to one UEs. Thus, the vector S_i can be collapsed to a single variable. To create the desired pattern, we define, then, $S_i{=}(F_i, D_i)$, where $F_i{\in}\{I, D\}$ is the frame type, and D_i indicates the distance of the corresponding frame in the current GoP. The cognitive engine processes the state sequence contained in the repository to create the wanted pattern. The complexity and computation overhead of the FHTP algorithm is higher compared to FDTP as the cognitive engine in FHTP needs to store a number of states equal to the GoP length the state repository.

3.10 Simulation Results

First, we show the optimal policy for a case with one UE. For simplicity, we fix the burstiness parameter and only control the transmission probability in the reference and differential frames. Fig. 3.15 shows the optimal transmission probability as a function of the weight

LTE Parameters	**Value**
MAC Scheduler	Proportional Fair (PF)
RLC mode	RLC UM
eNodeB power	25 dBm
UE Tx power	15 dBm
D2D Tx Power	7.5 dBm
Antenna Model	SISO
Path Loss model	Friis Propagation Loss
Uplink EARFCN	18100
UE-eNodeB distance	200 m
eNodeB - D2D distance	1500m
D2D nodes distance	20 m

Table 3.1: LTE parameters for NS-3 simulation.

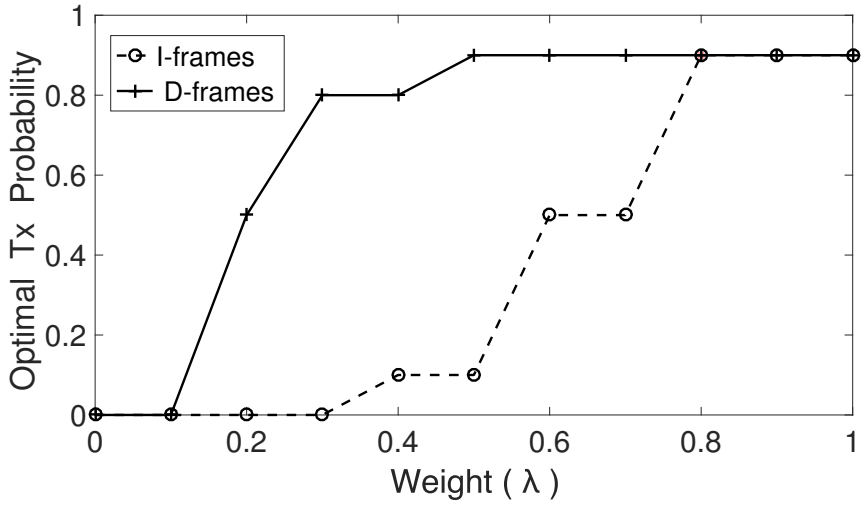


Figure 3.15: Optimal transmission probability as function of wight λ. The transmission burst size is taken as fixed = 2.

λ. As expected, the learning framework selects idleness when λ is equal to 0 and then progressively increases the transmission probability in the differential frames as λ increases. At saturation, the optimal policy then activates transmission during in the reference frame. Note that the same structure of the policy was shown in [34]. Fig. 3.16 shows how the accumulated reward value of the primary and secondary users follow opposite trends with respect to the variation of weight λ based on the reward function mentioned in equation [4].

We compare the policies proposed herein against a simple scheme where the D2D node predetermines transmission power and access probability ρ to induce a fixed average failure probability to LTE packets. Then, in each slot the D2D terminal transmits with probability ρ with a predetermined transmission power. We refer to this strategy as Fixed Probability (FP). Note that such policies could also be the results of imperfect SINR control, where packet loss is spread across the stream.

We assess the performance of the proposed schemes in the case-study scenario described in Section 3.9 by means of detailed network simulation. We use NS-3 version 3.23 [10] with LTE protocol stack to simulate the considered D2D and LTE scenario. In our simulations, D2D shares a portion (set of sub-channels) of LTE uplink spectrum. First, we packetize the video

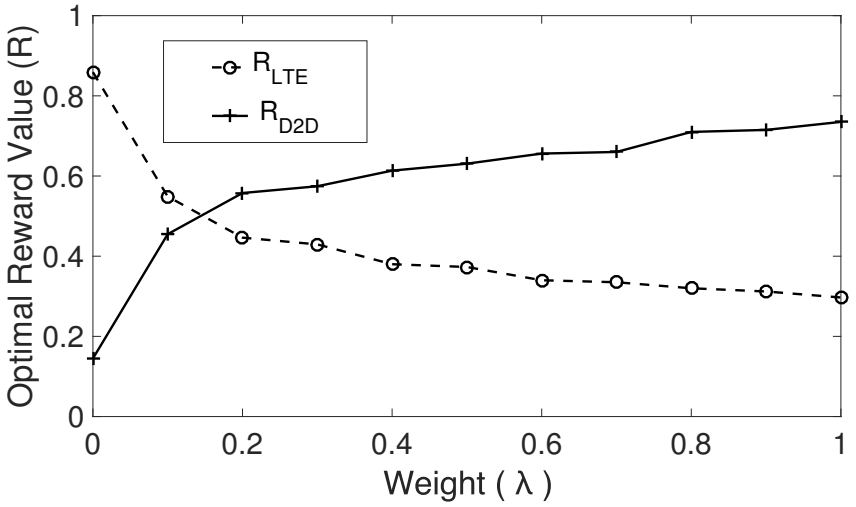


Figure 3.16: Optimal accumulated reward as function of wight λ.

with ffmpeg to generate the transport streams, which are input to the NS-3 simulator with a full end-to-end topology over the LTE protocol stack. The packet loss pattern is applied to the received video, which is, then, decoded and fed to the object detection algorithm implemented in MATLAB. The parameters used in the experiments are reported in Table 3.1.

As we consider real-time delay sensitive application, we disable Hybrid Automatic Retransmission Request (HARQ) and use RLC Unacknowledge Mode (UM). We will explore the effect of HARQ strategies on the proposed framework and video processing scenario in future work. The interested reader can find in [32] a discussion and evaluation of HARQ and video streaming. We use the channel fading trace Extended Pedestrian A model (EPA) on NS-3 as indicated in the 3GPP standard. To simulate the application layer, we build a transport stream of the video composed of a sequence of packets of equal size 188 bytes, which we transmit via UDP transport protocol. First, we conduct experiments with one LTE UE sharing the spectrum with the D2D transmission. In this setup, we show the performance variation with respect to the network topology, transmission power and mobility. Then, we measure the performance in a scenario with multiple active UEs, where the D2D link is assigned a subset of LTE frequencies.

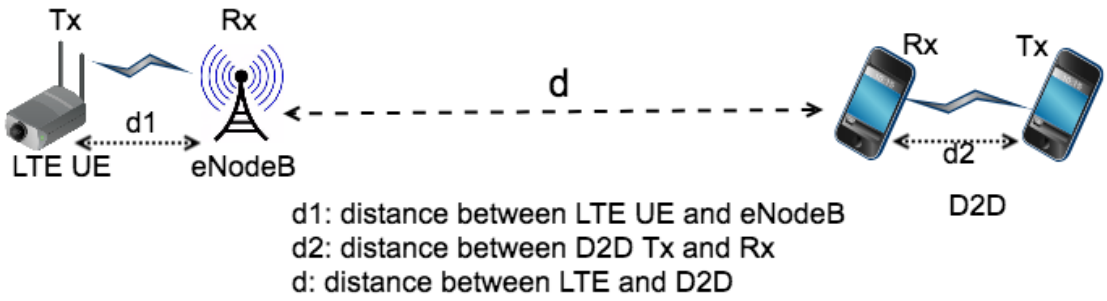


Figure 3.17: Topology of the LTE and D2D links considered in the results.

Topology: Fig. 3.17 illustrates the line topology used in the simulations. The distance between the UE and eNodeB and the D2D transmitter and its receiver are $d1$ and $d2$, respectively. We keep $d1$ and $d2$ fixed and vary the distance d between the eNodeB and the D2D receiver. First, we create a low-interference regime between the D2D and the LTE links by setting the topology parameters to obtain an average LTE video packet loss of 4% when the D2D link is active. Then, we reduce the distance d to create a high interference regime with average LTE video packet loss of 9% when the D2D link is active. For these two regimes, we obtain curves corresponding to different transmission probabilities of the D2D user. Fig. 3.18 plots the resulting object detection rate as a function of the D2D link's throughput. It can be observed that, in the high interference regime, the baseline protocol (FP) achieves object detection rate below 0.5 even with D2D throughput less than 10%, and then sharply decreases as the D2D link throughput increases. The low interference regime results in a less steep decrease, but still the object detection probability is heavily impacted as the D2D link becomes more active. The FDTP algorithm achieves a much smaller deterioration of the object detection rate, which saturates at approximately 0.4 with a 30% gain with respect to FP in high interference regime. For a given throughput, the past dependent FHTP achieves a performance gain of 35% over FP, and a perceivable gain compared to FDTP. In the low interference regime, FDTP and FHTP outperform FP by approximately 60% and 70% on average, respectively.

Transmission Power: We perform an analogous experiment, where we define the low and

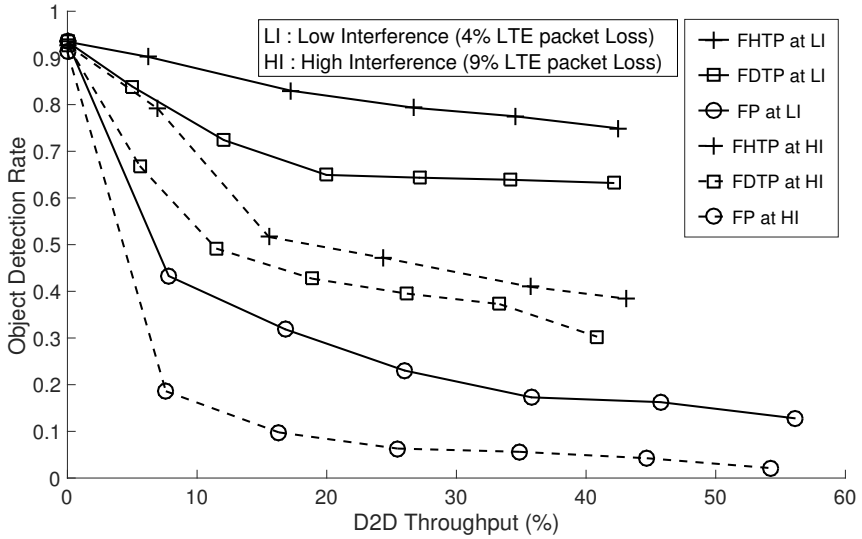


Figure 3.18: Object detection rate as a function of the throughput of the D2D link. The points in the curves are obtained by changing the transmission rate of the D2D transmitter.

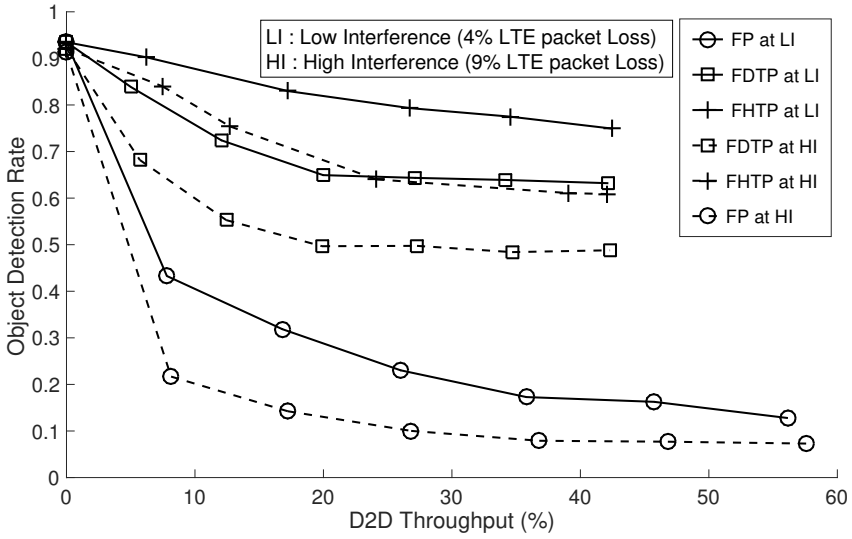


Figure 3.19: Object detection as a function of the throughput of the D2D link. The points in the curves are obtained by changing the transmission rate of the D2D transmitter.

high-interference regimes by varying the transmission power of the D2D link instead of the topology. The two interference regimes have the same packet loss probability at the LTE receiver given that the transmitter is active or idle. It can be observed that for any fixed throughput all the protocols improve the achieved object detection compared to that obtained by changing the topology. However, both FDTP and FHTP further increase the gap with respect to FP, which amounts to 45% and 60%, respectively.

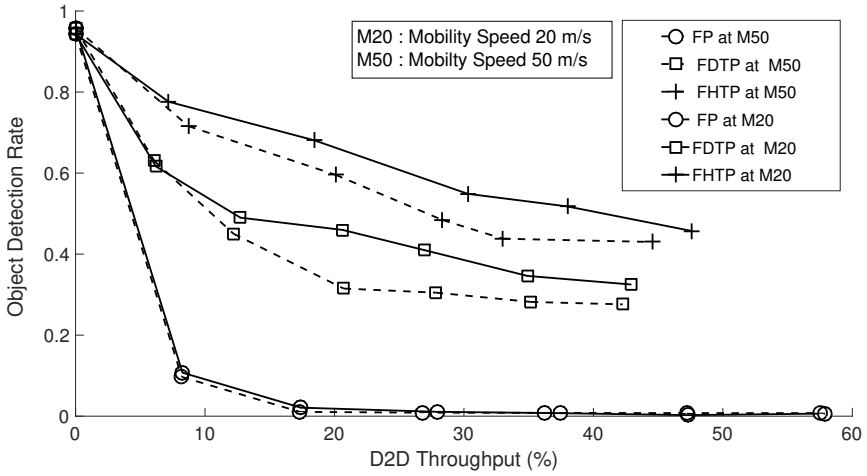


Figure 3.20: Object detection as a function of D2D throughput in presence of mobility. The points in the curves are obtained by changing the transmission rate of the D2D transmitter.

Intuitively, for a set of fixed parameters, a reduced distance between the D2D and LTE links increases the interference at the D2D receiver, thus reducing the achieved throughput of the D2D link, whereas, increasing D2D power to achieve high interference regime will enhance the D2D throughput. Hence, for a given D2D throughput the LTE throughput will be boosted in the latter case.

Mobility: We measure the performance of the object detection algorithm in a setting where users change their positions over time. In particular, the D2D node moves according to a random walk mobility model in a 50m×50m grid with uniform speed of 20m/s and 50m/s respectively. Fig. 3.20 shows that all the protocols are affected by mobility, with a increasing degradation of the object detection rate as the speed is increased. However, even with high mobility, the FDTP and FHTP transmission schemes achieve a gain of 40% and 50% over FP, respectively.

Object Recognition Accuracy: As mentioned earlier, we include in our numerical evaluation object recognition based on Gaussian mixture model from the MATLAB Computer Vision System Toolbox. The accuracy of the classifier depends on the number of mixture components, foreground noise during segmentation and also the size range of the bounding box

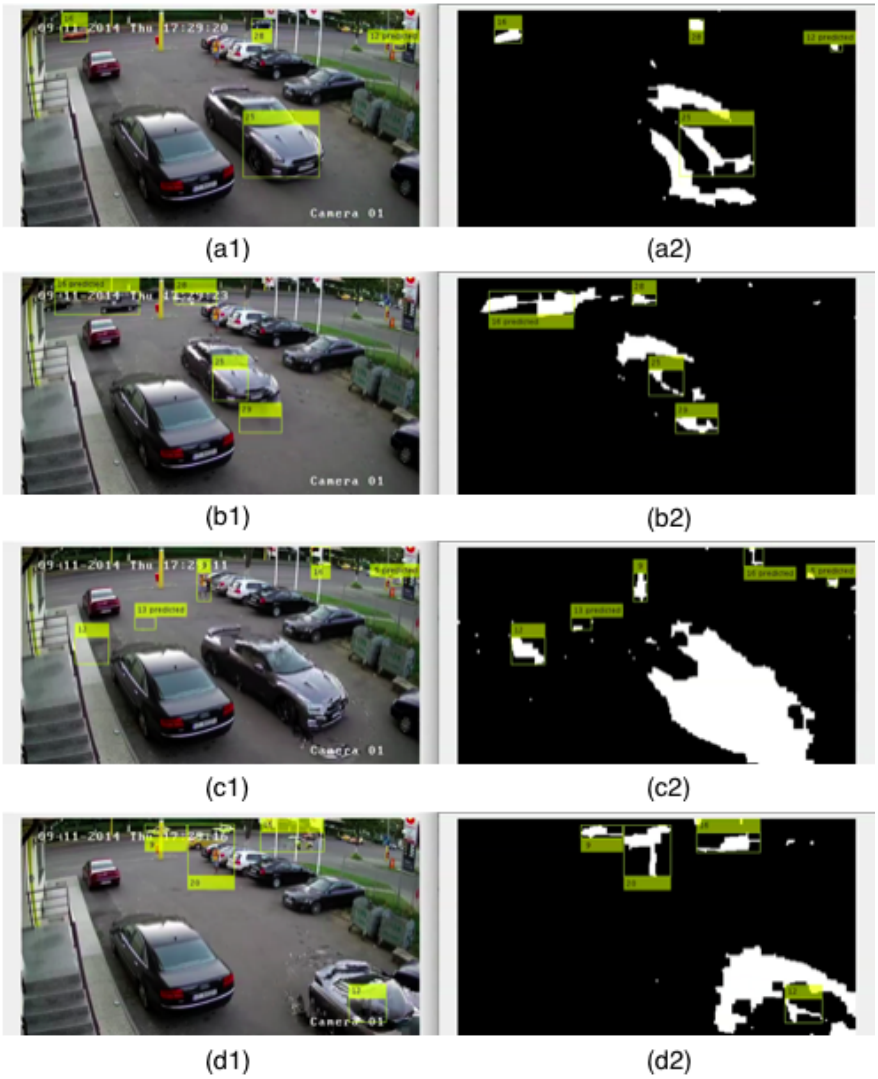


Figure 3.21: Selected frames from Object recognition of video where objects are detected based on background separation and classified based on a pre-trained model.

defining the objects.

Fig. 3.21 depicts the output of the algorithm for the recognition of moving objects, which, based on training from previous video frames, excludes static background objects by segmentation. In Fig. 3.21.a1, b1, c1 and d1 show frames from the video and a2, b2, c2, d2 are corresponding foreground masks obtained by background separation. Fig. 3.21.a1 and a2 show the correct object detection. In Figure. 3.21.b1 and b2, it can be noticed that a shadow of the moving car is detected as a separate object which indicates a false positive.

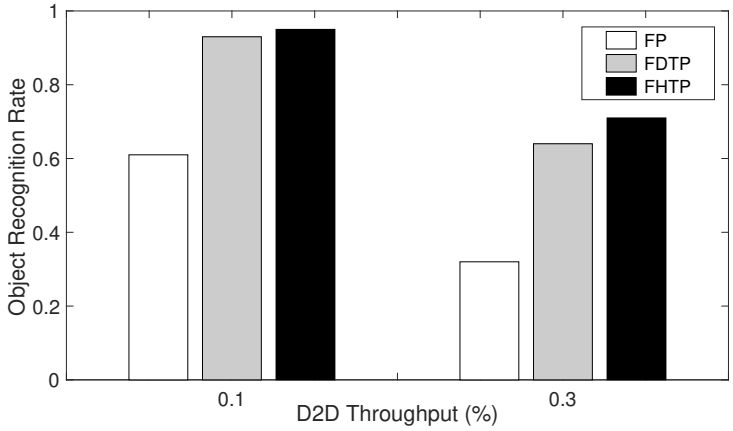


Figure 3.22: Object Recognition Rate

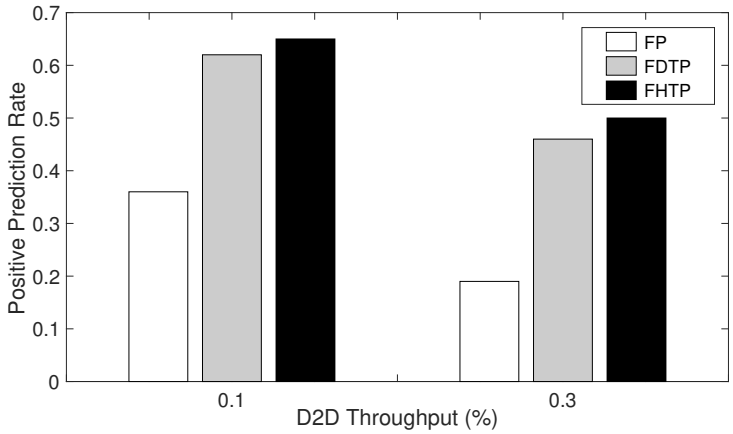


Figure 3.23: Positive Prediction Rate

In Fig. 3.21 c1 and c2, the algorithm detects two false positive objects where objects are not present and also one false negative as it does not detect the moving car in the middle. In Fig. 3.21.d1 and d2, the algorithm recognizes all the moving objects. However, due to corruption in the image (near the tail of the moving car at the bottom of the image), the centroid is shifted.

We measure the accuracy of object recognition from lossy video in an interference regime where D2D transmission causes 4% packet loss at the LTE receiver. We chose a topology corresponding to 10% and 30% D2D link throughput and measure the object recognition metrics achieved by FP, FDTP and FHTP.

Fig. **??** depicts the object recognition for the FP, FDTP and FHTP protocols. The large

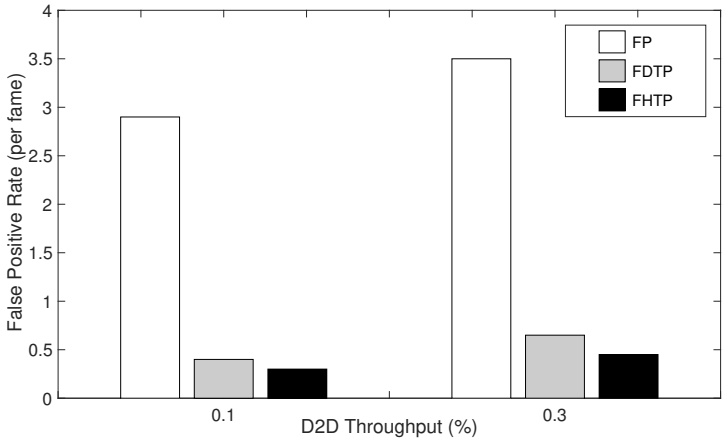


Figure 3.24: False positive rate

gain granted by FDTP and FHTP with respect to FP is apparent in both the settings. Fig. **??** shows that the positive prediction rate is also significantly improved by FDTP and FHTP compared to FP. In both cases, the absolute value of the recognition rate and the positive prediction reduces with the increase of D2D throughput due to the larger packet loss affecting the video stream. Fig. **??** depicts the false positive rate per frame. Again FDTP and FHTP outperform FP, as the corruption of frames due to the larger effective interference associated with packet loss in reference frames induces the detection of non-existing objects in the foreground. In all cases, the introduction of memory in the system cognition of FHTP improves the performance of the algorithm.

Efficiency: The efficiency measure defined in Sec. 3.9.1 is a function of the D2D link transmission probability and power. Intuitively, increasing the transmission power or the transmission probability of the D2D link increases its achieved throughput. However, it also results in a larger degradation of the object detection rate of the video over LTE. In results not shown here due to space constraints, we found that in order to achieve high efficiency, the FP protocol needs to drastically reduce the transmission probability of the D2D link for any value of the transmission power. Consequently, high efficiency is obtained in a region where the throughput achieved by the D2D link is small. Conversely, FDTP and FDHP maxi-

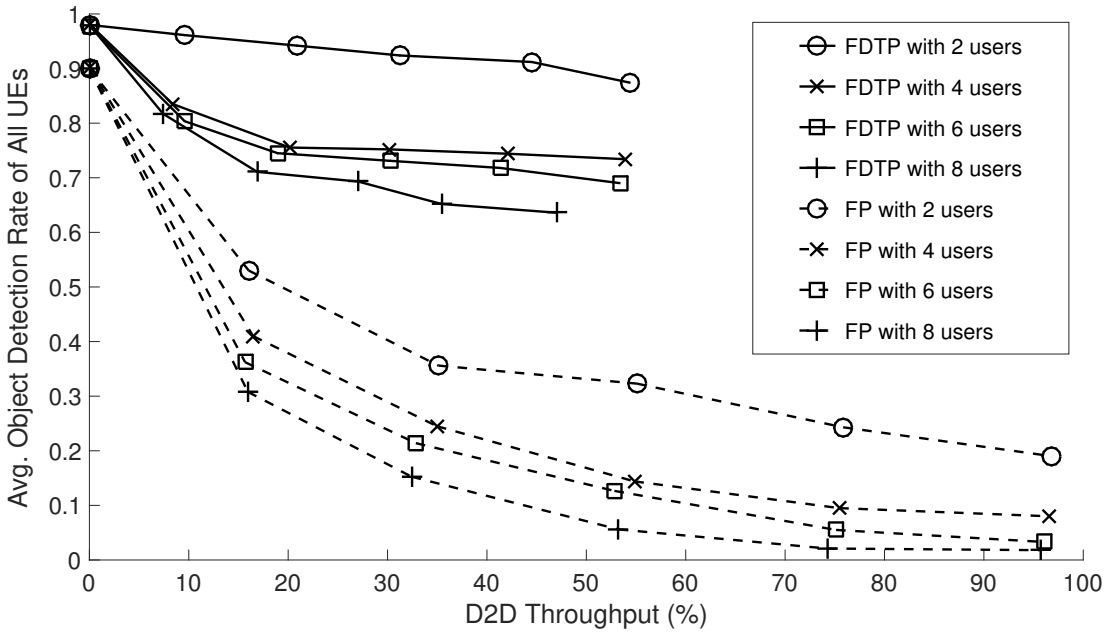


Figure 3.25: Object detection as a function of D2D throughput with varying number of LTE UEs when D2D uses fixed 6 PRBs. The points in the curves are obtained by changing the transmission rate of the D2D transmitter.

mize the efficiency when transmitting with high probability, but relatively small power (6-8 dBm). Thus, the maximum efficiency is achieved in regions where the D2D link occupies large portions of the channel while collecting throughput even when transmitting with low power thanks to its compact topology.

Multi-UE scenario: First we assign a fixed resource pool (6 PRBs) to the D2D link and vary the number of LTE UEs. We positioned the UEs in a uniform disc and adjusted the initial transmission power of the UEs to have a maximum packet loss equal to 3.5% when the D2D link continuously transmits. Fig. 3.25 shows the resulting average object detection rate across the UEs. It can be seen that as the number of UEs increases, the average object detection probability decreases. However, the FDTP gain with respect to FP is about 40%. Noticeably, the absolute value of the D2D throughput decreases as the number of UEs increases due to the larger probability of having at least one sub-channel allocated to an I frame. To illustrate this effect, in Fig. 3.26 we depict the allocated transport block (TB)

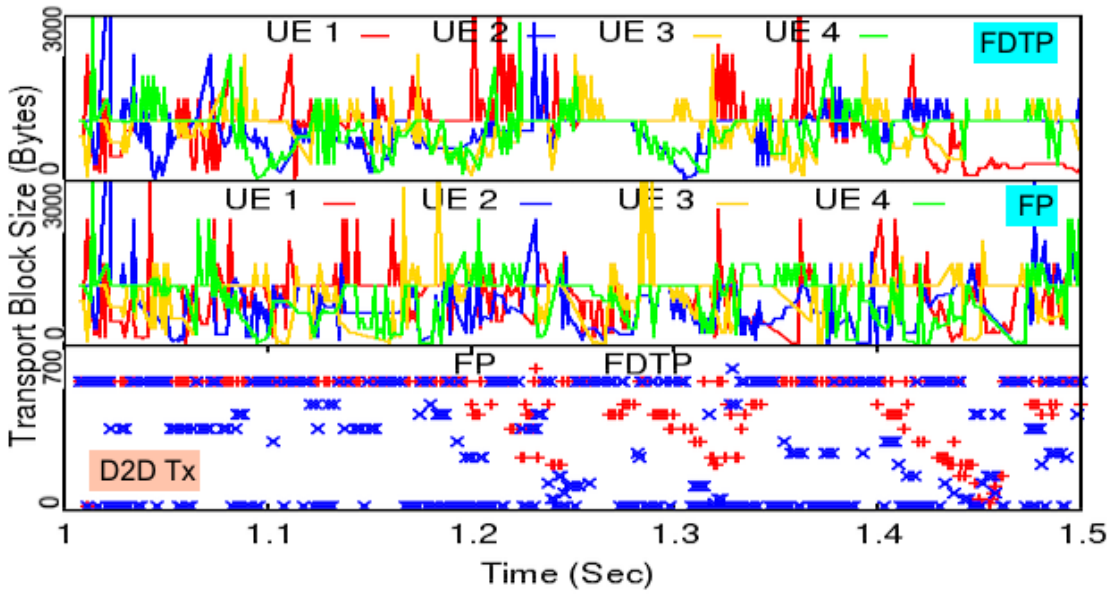


Figure 3.26: Transport Block allocation map for 4 UEs and D2D in FP and FDTP. The plot is obtained when D2D always transmits when all of the UE has differential frames.

size for 4 UEs and the D2D terminal when the D2D transmits with $\rho = 1$. The bottom plot indicates that the D2D transmitters backs off more if FDTP is used compared to FP, as the former does not transmit if any UE is transmitting an I-frame transmitted.

We also studied the effect of the size of the resource pool assigned to the D2D transmitter. For a setting with 4 LTE UEs, we tested a resource pool size equal to 6, 15 and 25 RBs, respectively, and measured the average object detection rate achieved by FDTP and FP as shown in the Fig. 3.27. FDTP achieves about 40% gain over FP, and the average object detection rate decreases as the resource pool size increases.

3.11 Related Work

A significant body of recent research exists which addresses D2D communications over cellular [27] and LTE networks [133]. The notion of cognition has also been studied to alleviate spectrum sharing issues in D2D communications using various resource allocation techniques [65, 218, 35, 34].

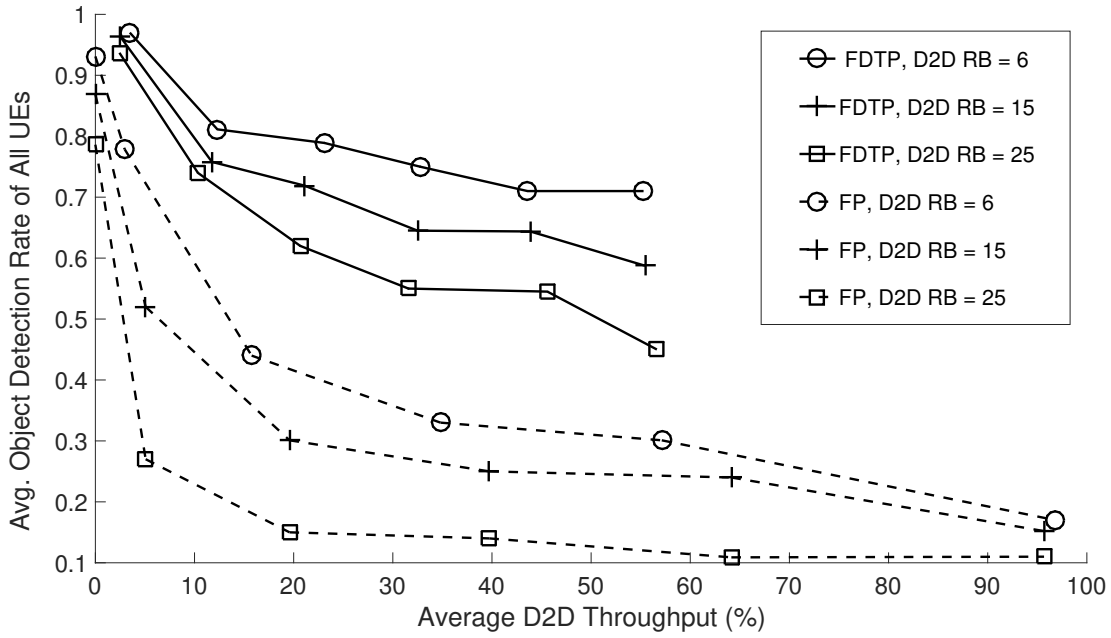


Figure 3.27: Object detection as a function of D2D throughput with 4 LTE UEs and varying D2D resource pool sizes. The points in the curves are obtained by changing the transmission rate of the D2D transmitter.

Interference control in D2D underlaying LTE: Several techniques have been proposed to mitigate mutual interference between LTE uplink and D2D communications including differentiated scheduling and QoS-aware scheduling. In [74], Doppler *et al.* proposed a mode selection mechanism where the D2D transmitter device selects a mode of communication in a set, *e.g.*, using shared spectrum or dedicated spectrum or communicating using the cellular infrastructure. It is shown that learning the strategy even in a single cell environment can improve mode selection criteria in multi-cell environments. Another study investigated mode selection via a relay node [134], where the D2D link performs transmission either using uplink overlay or underlay to control interference. A distributed power control mechanism was proposed in [84] which uses path loss measurements to control the transmission power of D2D devices. All these studies aimed at optimizing the network performance based on the overall network utilization, but did not specifically consider the performance of real-time services.

Coexistence of wireless technologies: Recent paradigms, *e.g.*, LTE-Unlicensed (LTE-U), Li-

censed Assisted Access (LAA) [30], propose LTE transmissions in unlicensed bands. The coexistence of LTE with other technologies with non-centralized architectures, such as WiFi, in unlicensed bands, poses several challenges, whose solution requires sophisticated techniques to control mutual interference. Qualcomm proposed a technique called Carrier Sensed Adaptive Transmission (CSAT) [6], where LTE operates with an adaptive duty cycle. The LAA standard uses an energy detection-based Listen-Before-Talk (LBT) mechanism [113], which is coupled with Clear Channel Access (CCA). Another technique called Almost Blank Subframe (ABS) was proposed in [17], which reduces the power and activity in selected downlink subframes to favor coexisting WiFi transmission in the same spectrum. The goal of all these protocols is to improve overall network throughput by maintaining the fairness of LTE and WiFi access.

Adaptive Video Transmission: Adaptive video transmission mechanisms, *e.g.*, HTTP Live Streaming (HLS) [188] and Dynamic Adaptive Streaming over HTTP (DASH) [187] were developed in recent years to provide reliable video transmission against variations in the capacity of wireless links. However, these protocols are based on TCP, which may incur an excessive delay in real-time applications. Moreover, these techniques are most suited to the streaming of stored video, due to the need to create multiple versions of it with different quality. Moreover, current video processing algorithms assume a fixed quality and resolution of the input. Finally, adaptive GoP can dynamically vary the compression rate. However, it would likely create highly compressed segments missing features important to achieve high quality of the output of the processing algorithm.

Edge-assisted Computation for real-time applications: The interested reader can find an in-depth discussion of the challenges of real-time applications, *e.g.*, video streaming, over wireless networks in [81]. To address these issues in complex urban IoT and Smart City environments, applications such as smart transportation (*e.g.*, traffic monitoring) and surveillance systems [55] adopt edge-computing solutions. In this context, edge-assisted architectures for

video surveillance over wireless networks were proposed in [33].

3.12 Conclusions

In this study, we proposed a content-based cognitive transmission strategy for hybrid D2D/LTE networks supporting urban IoT applications. A monitoring application was considered, where video data streams are remotely processed to detect relevant objects. The proposed cognitive strategy shapes the transmission strategy to match the structure of video encoding and processing. Numerical results show significant throughput gain of the D2D link for a given performance degradation of the monitoring application with respect to the case when the interference process is fixed throughout the video transmission. In future work, we will consider more articulated scenarios, such as object detection from a multiple smart cameras, more sophisticated computer vision algorithms and intelligent HARQ mechanisms which can further improve real-time applications for public safety and monitoring.

Chapter 4

Network Function Virtualization for Real-time Edge Computing

4.1 Introduction

In the Internet of Things (IoT), interconnected sensors collect and transmit data for analysis to remote servers [114]. However, the delivery of content-rich data may incur delay or loss due to the limited network resources. Indeed, network congestion may impair the ability of the system to support real-time services, such as video surveillance, traffic monitoring, smart transportation or the virtual reality (VR) and augmented reality (AR) [16]. For this family of applications, the cloudlets [198] and edge computing [103] paradigms mitigate the issue of latency by placing computation-capable devices within a one-hop wireless topology. However, in many network scenarios of interest, the coexistence of these demanding data streams with other services over constrained wireless networks necessitates new technical solutions.

Recent frameworks based on Software Defined Networks (SDN) [160] have demonstrated the

ability to improve network resource management using dynamic flow control and Network Function Virtualization (NFV) [97]. At the communication level, Software Defined Radios (SDR) [25] have been used to dynamically adapt the parameters of wireless transmissions. However, the main challenge of effectively utilizing the available bandwidth to support real-time applications producing large volume traffic stands. To this aim, the notion of Quality of Computing (QoC) [33] has been recently proposed to relax interference constraints on IoT data streams and facilitate their coexistence.

In this work, we propose a computation-aware communication control framework for real-time IoT applications generating high-volume data traffic processed at the network edge. Driven by QoC requirements, the framework provides real-time user-controlled packet replication and forwarding inside the in-kernel Virtual Machines (VM) using an extended Berkeley Packet Filter (eBPF) [44]. The implementation uses the concepts of SDN and NFV to achieve highly programable and dynamic packet replication. Resource allocation is semantic and content-aware, and, in the considered case, informed by the structure of data encoding. Numerical results are provided from real-world experiments demonstrating the enhanced adaptability and efficiency of the proposed solution.

4.2 Edge Computing for Real-time Applications

One of the core concepts used in this work is edge computing, which can reduce the computation latency of real-time applications. It is generally assumed that the edge servers are more powerful compared to the sensors, but offer inferior performance compared to cloud servers in terms of computation capabilities. We consider an application scenario where video data from different video sensors (cameras) are sent to the edge server, which runs in parallel several real-time processes analyzing the streams as shown in Fig. 4.1. Each computation process may use partial data from multiple sensors to accomplish its task. For instance, a

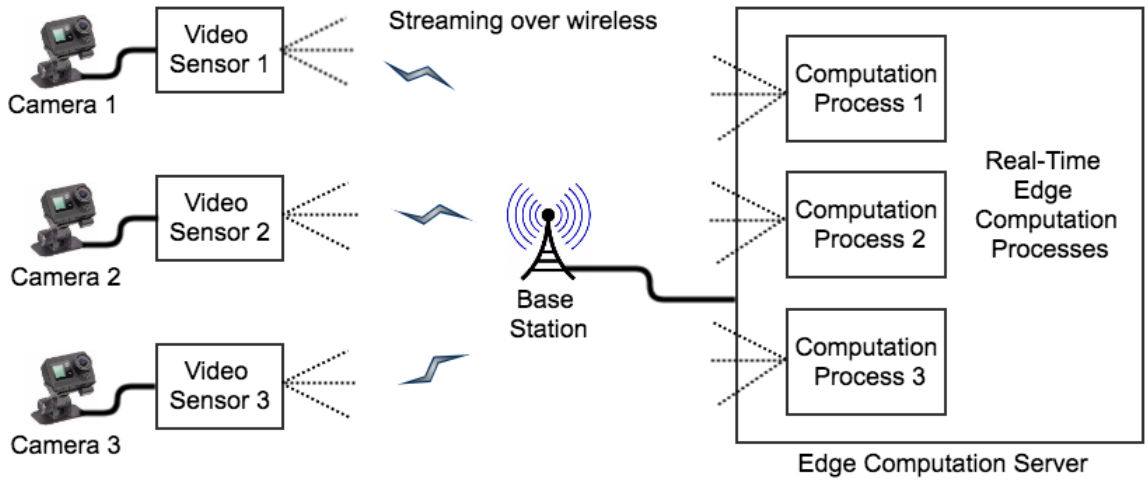


Figure 4.1: Real-time edge-based scenario: each computation process uses partial data from various sensors.

computation process aimed at fine-grained object detection may require high quality video whereas coarse-grained object detection (e.g. object counting) can operate on relatively lower quality and lossy video. We remark that this setup corresponds, for instance, to Urban IoT scenarios and, in particular, city monitoring.

Thus, in the considered multi-sensor multi-computation scenario, each sensor data might be reused at different system scales for different computational tasks. Note that if the sensors are mobile, the subset of sensors contributing to a specific computation process may also change over time. The "quality" of data a sensor should provide to a specific application is determined by the QoC requirements of the associated computation algorithm. Here, we define these requirements in terms of a set of metrics measuring characteristics of data delivery, such as loss and delay.

Let say, then, that at time T, N video sensors $\{S_1, S_2, ..., S_N\}$ are streaming video to M computation processes $\{C_1, C_2, ..., C_M\}$ running at the edge server. The i_{th} sensor data stream S_i is used by the j_{th} computation process C_j with quality Ω_{ij}, where $i = 1, 2, ..., N$ and $j = 1, 2, ..., M$. In a naive approach, each sensor S_i would transmit M streams of data over

the network with full transmission quality Q_i^{full}. Denote the bandwidth consumption stream S_i with quality Q as $S_i(Q)$ with B_i. The total bandwidth consumption for transmission with full quality is:

$$B_{total} = \sum_{i=0}^{N} \sum_{j=0}^{M} S_i(Q_i^{full}) = M \sum_{i=0}^{N} S_i(Q_i^{full}) \tag{4.1}$$

As an alternative approach, instead of sending M instances of $S_i(Q_i^{full})$ to M processes, the data streams are replicated in-network – that is, at the network edge – and sent to the M processes in order to reduce the network load to $B_{total} = \sum_{i=0}^{N} S_i(Q_i^{full})$. Thus, the number of streams is reduced to the minimum, but they are still transmitted at maximum quality Q^{full} to support any requirement the computation processes may impose. In the framework we propose, we further reduce the network load by dynamically tuning the quality of the data streams produced by the sensors to that required by the computation processes at any given time.

In the framework proposed herein, packets are replicated inside an in-kernel VM at the data link layer (layer 2) before forwarding the packet to upper layers. This results in a decreased use of memory at the upper layers of the network protocol stack. Our framework further reduces the system memory and CPU usage by incorporating content-based selective packet cloning, which is driven by the requirements of the individual computation processes.

In the light of these functionalities and objectives, in this work we address the following technical challenges:

i) Determine the quality of the data streams transmitted by each sensor at run-time.

ii) Select which of the data streams received at the network edge need to be forwarded, and to which computation process(es) at any point of time.

iii) Efficiently manage network and computation resources used by each computation process.

In order to solve the aforementioned challenges, we propose an approach based on SDN whose goal is to seamlessly control the resources of the network from the user space at the application layer.

4.3 Related Work

Several recent contributions focus on real-time IoT data streaming in edge computing architectures. A real-time video surveillance application is proposed in [34] where the transmission policy of the sensor is interference aware to facilitate coexistence with other communications. In [145], the authors adopt a stochastic optimization approach to maximize the efficiency of bandwidth usage by efficiently offloading computation to edge servers. However, the architectures presented in these works are not programmable. In [37], the authors present an edge-assisted SDN-based framework for the dynamic selection of the network used to transport data from real-time applications, and [63] proposes an SDN-based resource management framework to perform content caching and server selection. However, the aforementioned works do not address content reusability and flow orchestration framework at the network edge.

Some recent contributions proposed flow orchestration schemes using NFV at the network edge. Examples include NetFATE [135], a framework based on Open vSwitch (OVS) [164] running at the mobile edge. In [123], the authors proposed the platform NFVnice, which is built on OpenNetVM for the user space-controlled scheduling of NFV chains. However, while providing some level of flow control, these frameworks do not provide sufficiently low level control to extract in real-time for the content transported by the data streams and inform

content- and computation-based policies. Different from these contributions, the framework proposed herein adopts a content and computation-aware approach for resource management and dynamic control, and realizes an adaptable and programmable user-controlled platform.

4.4 Proposed Architecture

To develop the proposed framework, we use a built-in kernel feature called extended Berkeley Packet Filter (eBPF). In the following, we briefly introduce eBPF and, then, describe in detail the framework.

Packet filtering was first proposed by Mogul et al. [152] to provide user-controlled filtering of a subset of packets in the network stack of kernel which is known as CMU/Stanford packet filter (CSPF). The Berkeley Packet Filter (BPF) [147] has been developed to overcome the stack-based instructions and tree-based filtering model of CSPF. BPF employs a register based instruction set with directed acyclic control flow graph (CFG) for filtering mechanism which provides a significant performance gain over CSPF as CFG can retain the packet parsing states to avoid redundant comparisons. The BPF filter was implemented in Linux kernel 2.1.75 which supports 32 bit registers for instructions. However, a recent development of BPF called (eBPF) [93] in kernel 3.18 and above, further improves its performance by introducing ten 64 bit registers and employing the JIT (just-in-time) compiler. Also, eBPF programs can be invoked from different layers of the network stack, e.g. socket, qdisc, and drivers. This latter feature enables BPF to process the captured packets before forwarding them to the subsequent layer, and load user defined program inside the BPF in-kernel virtual machine to process the traffic dynamically. Also eBPF maps can be shared between the user and kernel space to enable seamless control. The recent development of BPF within the context of the open-source project IO Visor [85] resulted in the BPF compiler collection (bcc) with LLVM back-end and clang front-end. The main advantage is that the front-end

code can be written in high level languages such as python or C, which is translated in the bytecode by the BPF compilers to be loaded inside in-kernel virtual machine.

Based on these recent advancements, the schematics of the proposed framework are shown in Fig. 4.2, whose components are described in the following.

Quality of Computing Requirements: The QoS requirements of each computation process are continuously evaluated at the communication and computation layers performance over a temporal window. In the considered case, the QoC requirements are used to determine the minimum quality and portions of the data stream needed by the computation process. This parameter is reported by the computation processes to the application program, which updates the corresponding eBPF maps. The edge, in turn, reports it back to the corresponding sensor, which only transmits the required portion of the data stream and suppresses remaining portions to reduce the network bandwidth used by the stream.

Based on the determined requirements, the appropriate packets to be transmitted are identified by the sensor using a Deep Packet Inspection (DPI) module [33] at the application layer.

Layer 2 forwarding: Traffic forwarding to specific set of interfaces can be achieved by ingress/incoming and egress/outgoing port mapping through eBPF. In the proposed implementation, the user space application dynamically sets the ingress and egress port pairs in the eBPF map.

DPI eBPF Function: A DPI module is also implemented at the edge filters to select the packets to be forwarded to the computation process(es). This feature enables the implementation of content and computation-specific filtering policies. The DPI eBPF module runs inside the kernel, but can be accessed from the user space in real-time.

Selective Packet Cloning and Transmission: The function of this module is to reduce the

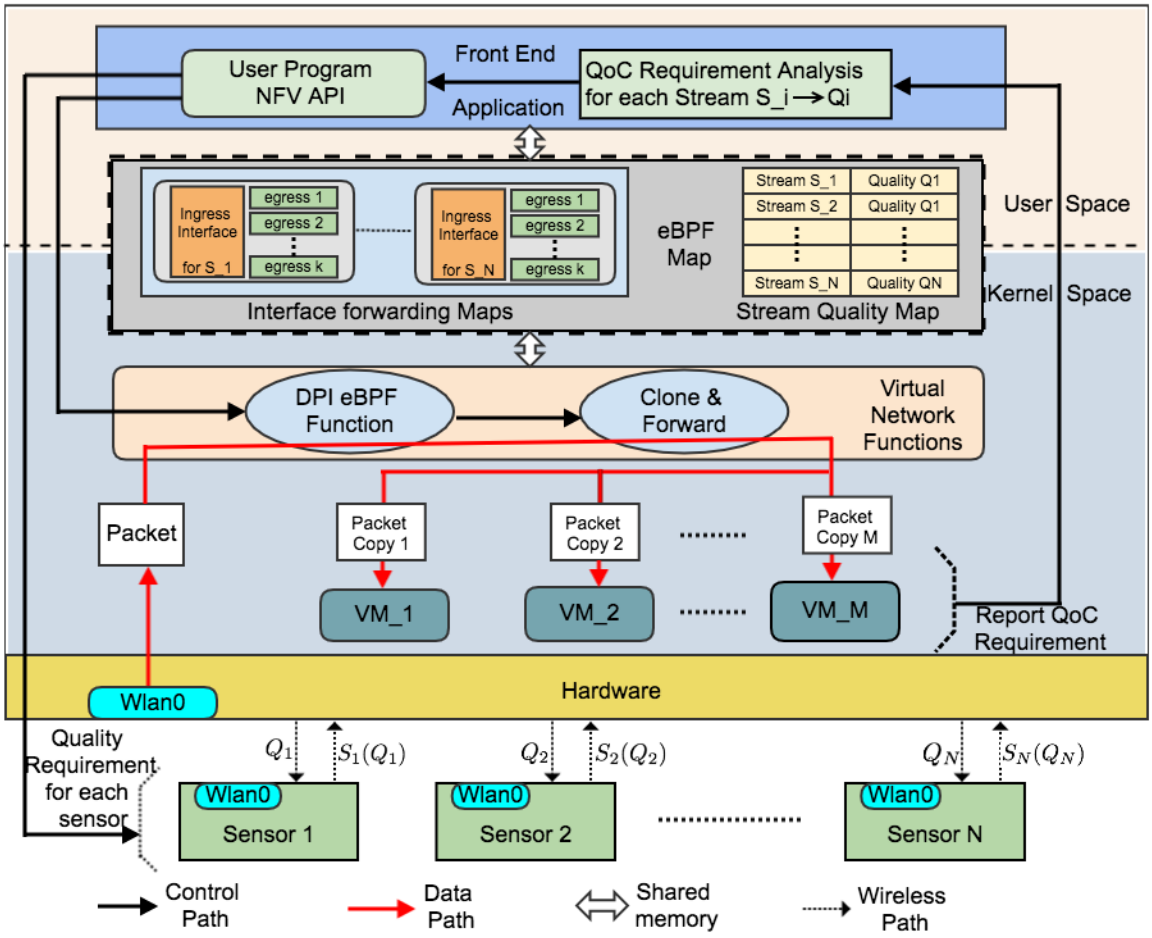


Figure 4.2: Schematics of the content- and computation-aware stream control framework.

usage of computation resources at the edge servers. To this aim, the module employs content-aware selective cloning of packets to be forwarded toward a specific computation process, thus minimizing the socket buffer allocation of reusable packets. The eBPF program running inside the in-kernel VM reads the QoC requirement for each specific computation process through the eBPF maps, and makes decisions in real-time regarding the cloning policy and packet forwarding.

4.5 Content and Computation-Aware Communication Control Protocol

In this section, we make specific the optimization problem and the control strategy integrated in the framework to the application scenario we consider. We remind that the objective of the multi-sensor and edge system is to minimize total bandwidth usage, while meeting the QoC requirements, that is, maintaining the QoC metric γ_j above a threshold δ_j for each computation process C_j. If each sensor uses a single stream S_i with transmission quality Q_i as described earlier, the optimization problem can be expressed as:

$$\min_{Q_i} \sum_i S_i(Q_i) \quad \text{s.t.} \quad \gamma_j \geq \delta_j, \ \forall i = 1, .., N; \ \forall j = 1, .., M. \tag{4.2}$$

To solve this problem, we first make the communication between the sensor and edge computation-aware. The sensor determines a transmission quality Q_i that satisfies Ω_{ij} for all $j = 1, 2, ..., M$. Thus, we define effective quality of stream S_i as:

$$Q_i^{eff} = \bigcup_{j=1}^{M} \Omega_{ij} \tag{4.3}$$

Intuitively, Q_i^{eff} is smaller than or equal to Q_i^{full}. The reduction in terms of network usage can be expressed as the difference between B_{total} and effective access network bandwidth B_{eff}:

$$B_{saved} = \sum_{i=0}^{N} S_i(Q_i^{full}) \ - \ \sum_{i=0}^{N} S_i(Q_i^{eff}) \tag{4.4}$$

We further reduce Q_i^{eff} by making the communication content-aware. In the context of the specific application scenario considered in this work, the video data stream is differentially encoded. As a result, the data stream is composed of packets transporting information relative to reference frames (full pictures) or differential frames (encoding differences with respect to reference frames). In [33], we provided a preliminary study on the effect of packet loss on the performance of object detection algorithms. Our work illustrates the much larger impact of packet loss localized in portions of the stream transporting reference frames compared to the loss of packets transporting differential frames. The solution we proposed concentrates interference on differential frames, rather than equally spreading it over the data stream.

Herein, we use this observation to build a content-aware packet filtering strategy. Specifically, when a reduction in the overall used bandwidth is necessary, the filter first drops packets that are transporting differential frames instead of uniformly dropping packets. Herein, we use a pre-built map, shown in Table 1, between the QoC requirement and packet loss in the two classes of content – that is, reference and differential frames. The advantage in terms of used bandwidth to achieve a given object detection accuracy is apparent. Object detection is performed using the Speeded Up Robust Feature (SURF) detection [11]. We summarize the steps of the filtering strategy and describe how content and computation-aware communications are implemented at the edge server and sensor side in Algorithm 1.

To implement the DPI eBPF function, and the cloning and forwarding function, we use Virtual Network Functions (VNF) which can run inside in-kernel VM. The VNFs can be invoked by the application program using APIs as shown in Listing 4.1. This allows the dynamic modification of the policies in the shared BPF maps. We further optimize the DPI VNF implementation by setting the MTU size as multiple of transport stream (TS) packet size (188 bytes); so that the module does not scan every byte of the UDP payload, rather,

Algorithm 1: Content & Computation-aware Communication

Input: $\mathbf{S} = \{S_i : i = 1, 2, ..., N\}$: Sensors
$\mathbf{C} = \{C_j : j = 1, 2, ..., M\}$: Computation processes
$\mathbf{\Omega}[i, j]$: Quality Requirement of S_i for C_j

Output: Sensor to Computation Map : $\mathcal{G} : S \mapsto C$
$\mathbf{Q}[n]$: Sensor Data Transmission Quality
$\mathbf{\Delta}[i, j]$: Edge Suppression Factor

Function EdgeControl $(\mathbf{S}, \mathbf{C}, \mathbf{\Omega})$:
  $\{I_i\}$: Interface list for S_i
  for $i = 1$ *to* N **do**
    $\{I_i\} \leftarrow$ **null**
    for $j = 1$ *to* M **do**
      if C_j *includes stream* S_i **then**
        $\{I_i\} \leftarrow \{I_i\} + I_j$
      else
        continue
    $Q_i^{eff} = \bigcup_{j=1}^{M} \Omega_{ij}$; $\mathbf{Q}[i] = Q_i^{eff}$
    for $k = 1$ *to* $|I_i|$ **do**
      A_k : Application on k_{th} VM
      δ_i : Packet loss tolerance of A_k for sensor data S_i $\mathbf{\Delta}[i, k] = \delta_i$
      $\mathcal{F}$: Frame type obtained by DPI of S_i for A_k
      if $\mathcal{F} \in$ *Reference Frame* **then**
        Clone and forward packets to interface I_k
      else
        Apply $\mathbf{\Delta}[i, k]$ suppression at interface I_k
        Clone and forward packets to interface I_k

Function SensorControl $(\mathbf{Q}[n])$:
  for $i = 1$ *to* N **do**
    α_i : Loss tolerance of S_i for transmission quality $\mathbf{Q}[i]$
    P_i : List of packets to be suppressed using DPI $S_i \leftarrow S_i - P_i$; Transmit S_i

jumps from one TS header of 4 byte size to the next TS header in the packet. This minimizes the number of instructions needed in the BPF. The implementation can integrate any future VNF accessible though a BPF API at the user program.

Listing 4.1: APIs for accessing eBPF VNFs

```
// Access eBPF functions through API
 bpf = BPF(src_file = "ebpf_vnfs.c", debug=0)
function_DPI = bpf.load_func("vnf_DPI", BPF.SCHED_CLS)
 ...
function_clone_forward = bpf.load_func (
                                "vnf_clone_forward", BPF.SCHED_CLS)
 ...
// Access and update eBPF shared Maps
pol_map = bpf.get_table("policymap")
pol_map[pol_map.Key(idx)] = pol_map.Leaf(arr, 0)
 ...
```

4.6 Performance Evaluation

In this section, we first describe the experimental setup, and then provide numerical results assessing the performance of the proposed framework.

Object Detection (%)	0.5	1.0	2.0	5.0
Uniform packet loss	0.95	0.84	0.46	0.1
Differential packet loss	0.99	0.96	0.74	0.4
	Packet	loss	(%)	

Table 1: Object detection as a function of packet loss when the uniform and selective packet dropping policies are used.

4.6.1 Experimental Setup

We implemented the eBPF program on a network edge server with 8 core CPUs. All the VMs run on QEMU hypervisor and we used the docker version 1.11.2 to define network containers. The implementation is based on Linux kernel 4.7. As sensor data stream, we used a real-world video of 640x360 pixels and 372 frames encoded in H.264/AVC format at 30 frames per second. The video is converted to transport stream (TS) with the ffmpeg tool and transmitted from the sensor to the edge over UDP.

4.6.2 Results

We evaluate the performance of the framework in terms of programability, layer 2 forwarding performance, bandwidth utilization as a function of the computation requirement, system resource usage and scalability. We tested the real-time redirection of traffic to the corresponding interfaces with respect to dynamic modifications in the mapping between the network source and destination ports, and the programmability demonstrated to be seamless.

Layer 2 forwarding performance: As discussed previously, the selective cloning and forwarding mechanism is implemented at Layer 2 using the DPI module at the edge server. The program is executed inside the in-kernel VM which runs with low level instructions. As a consequence, the flow does not incur significant delay traversing through the upper layers of the network protocol stack. Moreover, the layer 2 forwarding function in eBPF is connected to the *qdisc* of the kernel similarly to the Linux bridge. Hence, the eBPF switch achieves analogous layer 2 forwarding speeds as Linux bridge. We compare them by measuring their average throughput using the "packetgen" utility setting the packet size to 512 bytes. We further compare the two cases using TCP and UDP. Both eBPF and Linux Bridge achieve $\sim$ 1 Gbps throughput over UDP and $\sim$ 2.5 Gbps throughput over TCP (without TCP Segmentation Offload (TSO) and Generic Segmentation offload (GSO)).

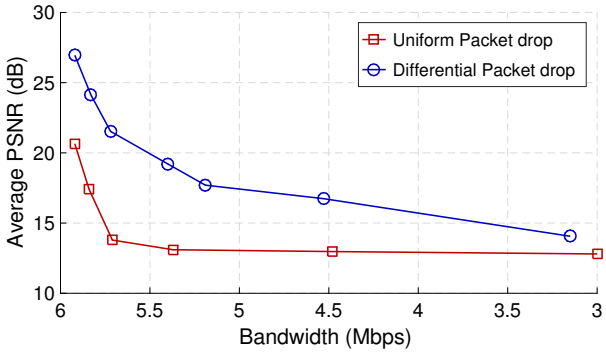


Figure 4.3: PSNR as a function of channel capacity.

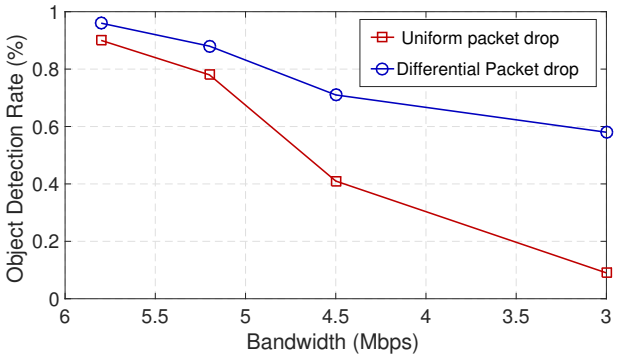


Figure 4.4: Object detection rate as a function of channel capacity.

Computation Performance: We assess the performance of the proposed framework in terms of quality of the received video with respect to the network bandwidth utilization. First, we measure the average Peak Signal-to-Noise Ratio (PSNR) of the video by suppressing packets at the sensor. Fig. 4.3 shows that as bandwidth usage is reduced by uniformly dropping an increasing number of packets, the PSNR decreases very sharply compared to the case where packet drop is focused on differential frames.

The DPI-based differential packet drop achieves significantly higher PSNR for any given bandwidth compared to the uniform, and non-selective, packet drop. For instance, when the available capacity of the communication link is equal to 5.5 Mbps, the selective filtering strategy achieves a PSNR equal to 18 dB, whereas the PSNR obtained using a uniform drop strategy is equal to 14 dB. A PSNR equal to 15 dB requires a capacity equal to 5.75 Mbps and 3.6 Mbps in the uniform and selective strategy, respectively.

Further, we assess the performance of SURF-based object detection with respect to channel usage (see Fig. 4.4). The selective strategy provides a significant gain in terms of object detection rate compared to uniform packet drop for any given channel usage. For instance, when the available channel capacity is equal to 4.5 Mbps, the object detection rate is equal to 0.4 and 0.7 in the uniform and selective drop strategy, respectively.

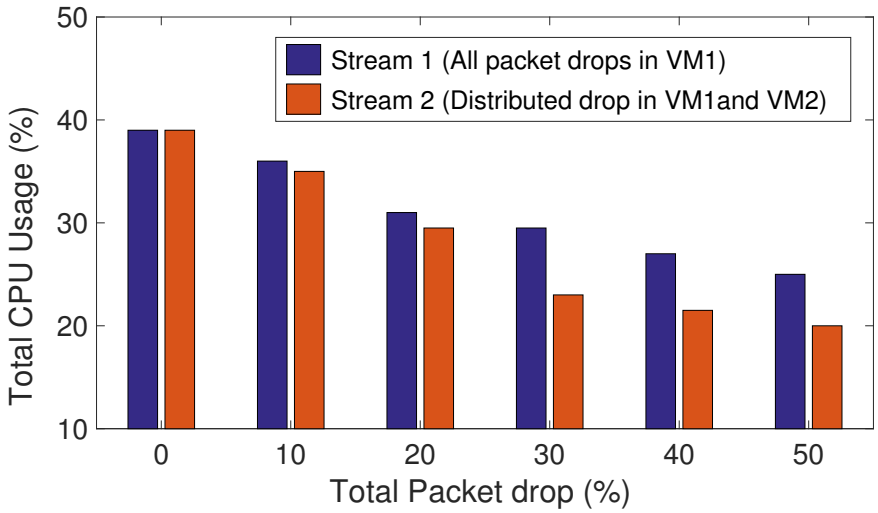


Figure 4.5: CPU utilization as a function of packet drop rate.

System Resource Utilization:

By selectively suppressing packets and avoiding cloning all the packets at the network edge, the proposed framework also improves system utilization. In the considered case-study, we use two video streams, each of which is processed by two processes running on two separate VMs (VM1 and VM2 respectively). We measure average CPU usage of all VMs with respect to packet drop rate. We distribute packet drop differently in the two streams: all the packets belonging to Stream 1 (first video) are dropped by VM1, whereas VM2 equally drops packets of Stream 1 and Stream 2.

Fig. 4.5 shows that the CPU usage is reduced by $\sim 3\%$ on average for every 10% total packet drop increase when all packet drops are performed on one VM. However, if packet drops are equally distributed between two VMs, then for every 10% total packet drop increase, the CPU usage is reduced by $\sim 5\%$ on average. This indicates that the proposed framework provides an increasing advantage as the differences in QoS requirements of the computation processes increase.

Scalability: In order to test the scalability of the framework, we create a large number of docker containers for parallel computations (see Fig. 4.6). On the sensor side, we use two

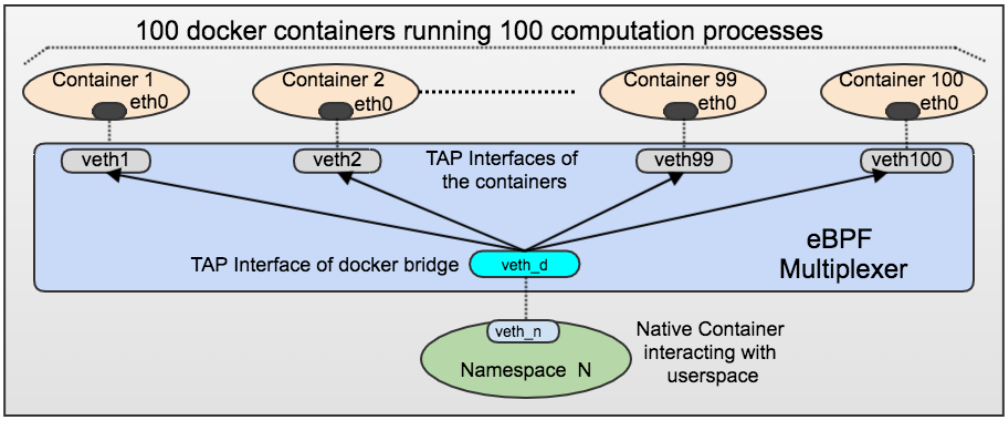


Figure 4.6: Multi-streaming test with 100 docker containers.

sensor devices streaming data. The main goal of this test is to assess whether or not the framework can handle the execution of many parallel instructions in the kernel. Our tests show that due to the light-weight implementation of the DPI and selective clone-forward functions, and the low-level execution of our framework, the framework performs adequately even when the number of computation processes is large. Fig. 4.7 shows that the total system utilization almost linearly increases with the number of containers. However, the slope decreases as the number of containers grows. This result indicates a tendency of the system resource utilization to become more efficient for large number of containers. Note that the total system utilization increases by $\sim 8\%$ for every 25% increase in packet drop rate, that is, the efficiency increases as the load increases.

4.7 Conclusions

The main contribution of this work is the design, implementation and test of an open-source, programmable computation-driven communication control framework for real-time edge computing systems using built-in kernel eBPF. The main features of the proposed system are: *(a)* reduced network utilization to support computation processes; *(b)* highly dynamic network and packet filtering control; and *(c)* efficient system resource utilization

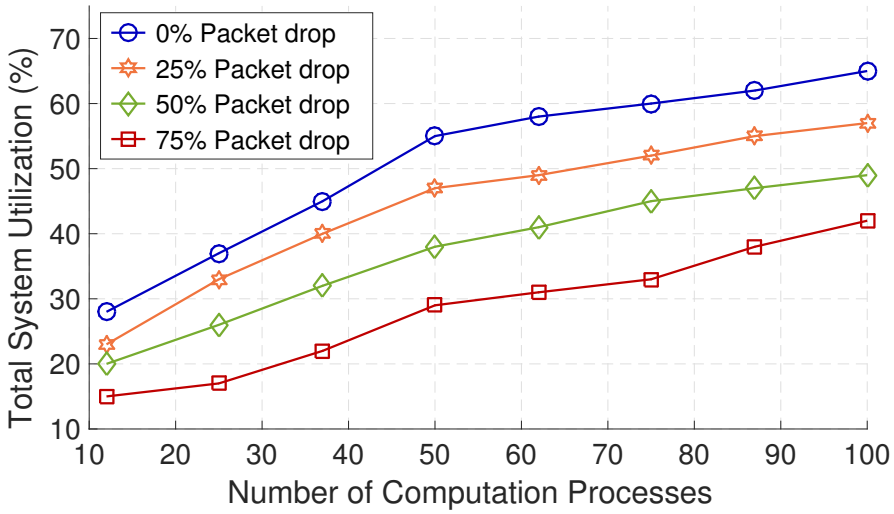


Figure 4.7: Variation of total system utilization vs number of containers for different packet drop percentage.

at the edge server. Although the framework is demonstrated in a specific application, that is, video processing, the architecture is flexible and can support a broad range of IoT applications. Numerical results obtained by means of real-world testing demonstrate the ability of the proposed system to dynamically adapt data streams supporting remote computation processes.

Chapter 5

Edge-assisted Content and Computation-Driven Dynamic Network Selection for Real-Time Computing Applications

5.1 Introduction

In Urban Internet of Things (IoT) systems [29] a large number of devices interconnect and interoperate to provide services to the citizens. Supporting this city-wide exchange of information using existing communication infrastructures is extremely challenging especially when IoT services coexist on the same network resource with traditional services, such as data and voice over cellular networks. Additionally, many relevant Urban IoT services, such as transportation and vehicular networks, are computation-driven, where data are processed in a, possibly multiscale, architecture to provide feedback and control signals in real-time.

The Quality of Service (QoS) constraints imposed by such services, besides being often more stringent than those where human users consume data, are specific to the computational-task and data structure.

To address this scenario, we introduce the notion of Quality of Computing (QoC), where the performance of the network is measured in terms of quality of the output of algorithms processing data. In this work, we propose a dynamic network selection mechanism based on Software Defined Networks (SDN) [160] designed to provide QoC in Urban IoT scenarios, where IoT data streams coexist with other services on the same network resource. The proposed mechanism dynamically assigns portions of the IoT stream to available licensed and unlicensed network resources to guarantee QoC while minimizing cost of operations and licensed band occupation. Instrumental to our technique is the recently proposed edge-computing architecture [89], where computational resources placed at the edge of wireless access networks enable the interconnection of network management to processing. Thus, data processing at the edge processor informs network management, which in the considered case takes the form of network selection for data stream portions with different relevance to the computational goal.

The framework and technique proposed in this work find wide applicability. Herein, we consider a real-time monitoring scenario, where sensors transmit a video stream which is analyzed at the edge processor to identify and classify objects. The supporting wireless infrastructure consists of a WiFi network, which operates in unlicensed frequency bands, and a cellular communication technology, Long Term Evolution (LTE) in our case-study, which operates in licensed bands. In our setup, uplink licensed LTE provides high QoC, where packets experience a "near interference-free" environment which comes at the cost of utilizing bandwidth that could be allocated to competing licensed services. Conversely, transmission over WiFi is assumed cost-free, but is subject to interference from other data streams, which may cause temporally localized congestion. In the proposed framework, the

edge-assisted SDN architecture dynamically selects the network resource to be used by the data stream, or specific portions of it, to trade-off licensed bandwidth usage and QoC.

We illustrate the performance of the proposed framework via experiments conducted on a real-world testbed emulating WiFi and LTE networks using hostapd [142] and OpenAirInterface [159, 158]. In the considered setup, QoC at the edge processor is periodically affected by bursts of traffic on the same channel. The SDN-based system we propose mitigates this effect by reacting to congestion and opportunistically moving parts of the data stream to the LTE network. To this aim, we leverage the specifics of spatial and temporal video encoding to separate the video stream into sub-streams with different relevance to the recovery operated at the edge processor. Numerical results show that the dynamic selection technique improves QoC while minimizing LTE usage.

5.2 Related Work

Recently, network selection techniques aiming at improving throughput and bandwidth utilization have attracted considerable attention. Stream Control Transmission Protocol (SCTP) [181], Concurrent Multipath Transfer (CMT) [108] and other analogous protocols have been proposed to select alternate network paths to improve reliability. Most of these innovative architectures are based on connection-oriented and stateful protocols, which often do not match the needs of real-time services. The Multipath TCP (MPTCP) protocol [204, 40] proposes to use multiple paths available in the network to build a single connection session, where packets are distributed across the paths proportionately to their respective quality. However, similar to TCP, the protocol prescribes retransmission and even an individual slow path may cause issues such as buffer overflow and delayed packet reordering at the receiver. This limits its applicability in time-sensitive applications. Network selection in multi-radio networks for IoT environments is proposed in [22] and [91]. However, the techniques pro-

posed in these papers are based on traditional throughput requirements and involve core network components with unpredictable propagation delays. Different from these previous contributions, we propose a framework where network selection is aware of the content transported by the stream and the global computation task. To this aim, our framework is edge-assisted, thus creating an innovative connection between content and network management, and avoiding the, possibly large and unpredictable, delay introduced by cloud- and core-network based architectures. Additionally, since network selection is controlled by the edge processor and local network managers, our technique minimizes impact on coexisting services. Specific content-based Interference control can be added to the proposed framework using approaches analogous to those in [34, 36].

5.3 Case-Study Scenario

Since processing architectures based on cloud-computing, such as [54], often cannot meet the stringent delay constraints imposed by real-time urban IoT services, herein we assume that data processing occurs at the wireless network edge. In particular, we assume that sufficient computational power, i.e., an edge processor, is available at one, wireless, hop from the sensors acquiring the data. We focus our study on a scenario where video input data are streamed by sensors over the wireless network to the edge processor, whose objective is to detect and classify objects within the captured area.

We assume that the overall bandwidth of the network infrastructure, which is composed of multiple individual networks and technologies, is sufficient to support the data stream. We focus on WiFi [192] and LTE [86] networks, which are widely available in urban environments, supported by most devices and predominant technologies for mid and long-range communications. The two networks have different characteristics. The LTE network has an access cost, which conceptually could even correspond only to the reduction in the bandwidth

available to other licensed users, but operates under Quality of Service (QoS) guarantees with dedicated uplink channel. The WiFi network, conversely, operates in the unlicensed band and does not incur an "operational cost", but is subject to congestion and interference from other, possibly bursty, data streams.

The video streaming application encodes the data using H.264 encoder [203] and then transmits using Real Time Protocol (RTP) [110]. The sequence of video frames is spatially and temporally compressed generating Group of Pictures (GoP), where each GoP contains a reference frame (full picture) followed by a series of differential frames, which transport differences with respect to the reference frame. Conversely, damage to a differential frame causes a quickly recovered corruption [34, 36], which is also mitigated by temporal interpolation. The encoded video frames are further divided into transport stream (TS) packets and encapsulated into the payload of data packets for video streaming.

5.3.1 Quality of Computation Metrics

Metrics such as packet latency and jitter have been traditionally used to measure the performance of video streaming for real time services. However, packet-based metrics which average over all packets may not fully capture the performance of the data stream, especially in relation to the output of the processing algorithm which is heavily dependent on spatially and temporally local properties of the received video. For instance, due to spatial and temporal encoding, if packet loss occurs within a reference frame, the resulting corruption spreads over the entire GoP creating persistent artifacts which may be erroneously detected as objects. Thus, the impact of packet loss, either due to packet delivery failure or excessive delay, is not uniform over the video stream.

A metric which is widely used to summarize video quality is Peak Signal to Noise Ratio (PSNR) [106], which measures the average Mean Squared Error (MSE) over all the video

frames using as reference the uncorrupted video, that is:

$$\text{PSNR} = 10 \log 10(K_{\text{bps}}/\text{MSE}_{\text{avg}}), \tag{5.1}$$

where K_{bps} is 2^B, B is the number of bits per pixel, and MSE_{avg} is the average MSE over all the frames of the video. However, two videos with same PSNR can have entirely different characteristics, especially for the purpose of computation.

Herein, we use a metric of quality which is directly dependent on computation. More specifically, we directly use correct object detection rate as a metric to measure the quality provided by specific packet loss patterns and guide network selection. Then, we define

$$P_{obj\_det} = \frac{N_{rcvd}}{N_{ref}} \tag{5.2}$$

where N_{rcvd} and N_{ref} are the total number of detected local features, corners and edge pixels of all matching surface objects in the received and reference video, respectively, such that $\{N_{rcvd}, N_{ref}\} \in N$ where N is the total number of local features of surface objects detected in the uncorrupted video.

5.3.2 Preliminary Study

In order to understand the impact of packet loss within the video stream on the quality of the computation output, we perform a numerical study on real-world video. Specifically, we generate synthetic packet loss patterns and measure the quality of the decoded corrupted video in terms of object detection. We first test "content-agnostic" packet loss patterns, where packets are dropped with uniform probability over the stream. Then, we test a "content-aware" pattern, where reference frames are protected and packets are dropped only in differential frames. In our experiments, packet loss is distributed according to an

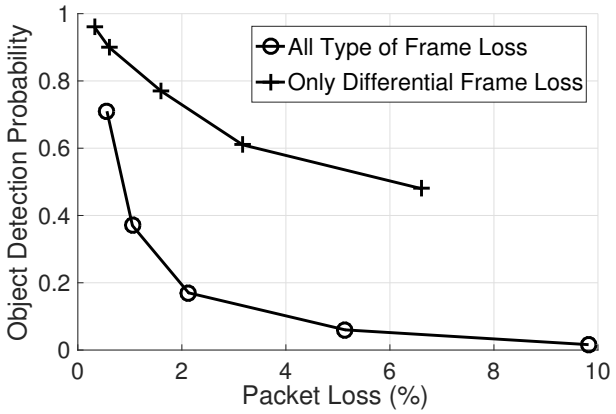


Figure 5.1: Object Detection vs Packet Loss

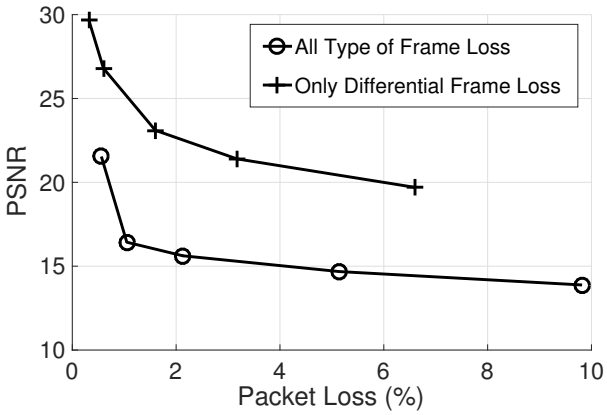


Figure 5.2: PSNR vs Packet Loss.

exponential distribution. Figure 5.1 depicts object detection probability as a function of packet loss for these two cases. It can be observed that concentrating packet drops in differential frame considerably improves object detection for a given percentage of packets drops. For instance, an object detection probability of 0.6 corresponds to 1% packet loss in the content-agnostic case and 4% packet loss in the content-aware case. Thus, content-aware packet loss patterns can tolerate 4 times the packet loss achieving the same QoC. Analogous observation can be made when measuring PSNR, see Fig. 5.2. We build our network selection framework on these observations, where the edge processor evaluates patterns of packet drops and informs network selection. Specifically, the edge processor matches recent packet loss patterns to QoC, and generates feedback used by the sensors to determine the network used to transport different portions of the video.

5.4 Architecture

The considered edge-assisted urban IoT architecture is composed of an edge processor connected to the WiFi and LTE base stations via a high-capacity link, and a set of sensors connected wirelessly. Figure 5.5 shows the components of the architecture and the flow of information. Sensors have two queues to maintain the state of packet flows on WiFi and

LTE, where one of the data paths can be used at any point of time for upstreaming data to the edge processor. The "state" of the data path associated with each individual network is based on feedback generated from the edge processor, which evaluates the pattern of received packets in relation to the specific computational goal. The output of the evaluation is sent to the sensors using reverse paths over all available networks in the form of feedback messages. This to avoid long delays and possible timeouts in case one of the two paths is currently congested, and improve reaction time of the system in selecting different networks in response to paths' state. Feedback messages are also used to probe paths, which is mostly effective when up and downstream links use the same channel resource. However, the consistency of feedback-based probing with the real state of the network is limited by the small size of feedback messages.

The edge processor connected to the base stations performs real-time computation and keeps track of recent QoC and packet loss patterns. Based on pre-computed mapping and the QoC requirement, the edge processor determines the packet loss threshold to be included in feedback messages. The sensor performs Deep Packet Inspection (DPI) to determine the content of packets, which informs network selection in response to the feedback and network state.

We make the architecture described above specific to the real-time video monitoring scenario at the center of our study. In this context, content differentiation corresponds to different types of video frames, as illustrated in the preliminary results discussed in the previous section. Once the sensor receives the tolerable packet loss limit, it determines the frame type of the packet to be transmitted and tunes the threshold for offloading the traffic to the LTE or WiFi network accordingly. In order to select the network, the sensor first measures the long term average packet reception quality at the edge from the feedback messages sent from edge. The edge sends application sequence number of the last packet received in every feedback to the sensor. Upon reception of every feedback message, the sensor updates

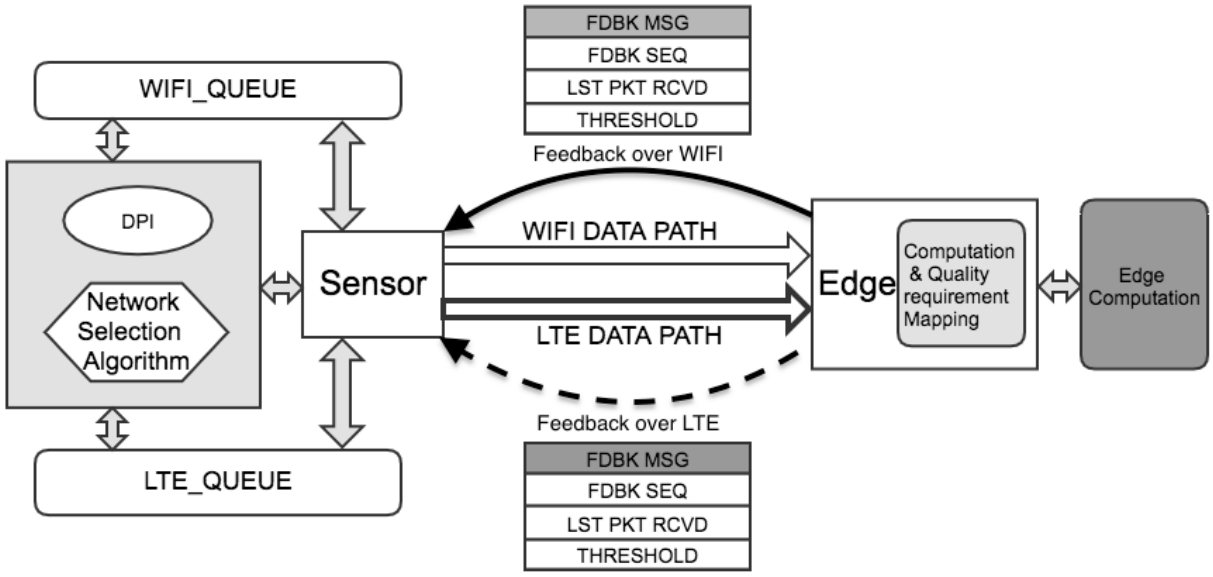


Figure 5.3: Edge Assisted urban IoT Architecture

the state queues for the WiFi and LTE accordingly. Then before selecting the network, the sensor reads the most recently updated queue and check if the corresponding network provides sufficient QoS to packets to match the threshold requirement. If so, it selects the current network otherwise, it offloads the traffic opportunistically.

The network selection algorithm also considers the cost of operation. To this aim, once data are offloaded to the LTE network, the protocol will try to re-route the traffic trough WiFi whenever a pattern is detected that can support the QoC requirement. Since the WiFi channel is bidirectional, performance measured using feedback roughly correspond to those of the upstream data. Thus, the protocol extracts useful information from the feedback, where good performance of the feedback channel is used to trigger a re-assignment of traffic to WiFi. Then, the protocol compares the sequence number of feedback message over WiFi with that that over LTE and, if they are comparable for a time window, it presumes the WiFi channel quality returned to a good state.

The basic functions of the network selection algorithm are summarized in Algorithm 2. We study two variants of the protocol, namely content-agnostic and content-aware protocols. In

the former case, the system treats all the packets in same way, i.e. uses a common packet loss threshold for network selection. In the latter case, the threshold value will be applied to the non-reference packets only, whereas the packets from reference frames will be transmitted using the LTE network, which corresponds to zero packet loss tolerance.

Algorithm 2: Content & Computation-aware Network Selection Protocol

```
   Input: Feedback control messages from Edge
 1 begin CollectFeedback () :
 2     update QUEUE_lte update QUEUE_wifi
 3     Δ_lte : Feedback delay over LTE
 4     Δ_wifi : Feedback delay over WiFi
 5     if  Δ_lte > Δ_wifi  then
 6         select QUEUE_lte
 7     else
 8         select QUEUE_wifi
 9 begin DeepPacketInspection () :
10     F : Frame type obtained by DPI of Tx packet
11     if  F ∈ Reference Frame then
12         packet_type = 1
13     else
14         packet_type = 0
15 begin NetSelect () :
16     Q = CollectFeedback()
17     type = DeepPacketInspection()
18     if  type == 1 then
19         Loss_thresh = T1
20     else
21         Loss_thresh = T2
22     if  |Q[start] − Q[end]| − Q_size < Loss_thresh then
23         if  Q == QUEUE_wifi then
24             Network ↦ WiFi
25         else
26             Keep same Network
27     else
28         Network ↦ LTE
```

5.5 Experiments & Numerical Results

We implemented the network selection algorithm on a fully functional testbed, where we use hostapd [142] to create software access point (AP) for WiFi and OpenAirInterface [159] as open-source LTE emulator. We use a real-world monitoring video of 641 frames, which are encoded with H.264 and converted to a TS packet stream using ffmpeg [46] codec. The TS packets are, then, encapsulated within UDP packets and transmitted over the network to the edge processor. To illustrate the general performance of object detection, we use Speeded Up Robust Feature (SURF) based object detection [43] as a measure of computation quality. The received video packets are first decoded using ffmpeg decoder and then fed into the object detection process.

5.5.1 Testbed setup

For the testbed emulation, we used laptop computers as sensor and edge machines. These machines run ubuntu 14.4 operating system with Linux kernel version 3.19. The WiFi software AP and also the LTE base station runs on the same machine as the edge processor to simulate a high capacity link. We configured the software AP using open-source hostapd software with configurations conforming with IEEE 802.11 standard. For LTE, we used open-source based OpenAirInterface (OAI) module installed on the machines that connect to radio interfaces USRP B210 with omnidirectional antennas separately for uplink and downlink. The sensor machine runs the code for LTE UE and the edge machine runs the LTE eNodeB code. We used licensed band for our experiment within allowable duration for academic research. As we consider edge computing, we do not use the EPC core network for LTE and only use the evolved universal terrestrial radio access network (EUTRAN) [137].

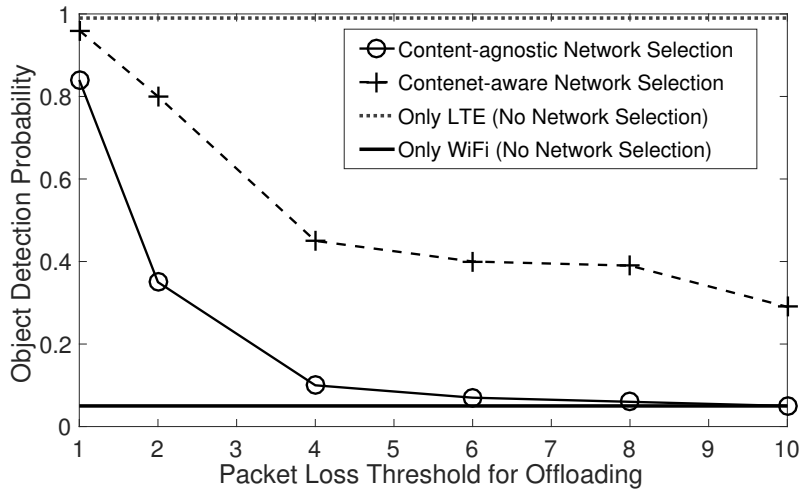


Figure 5.4: Object detection probability as a function of packet loss threshold

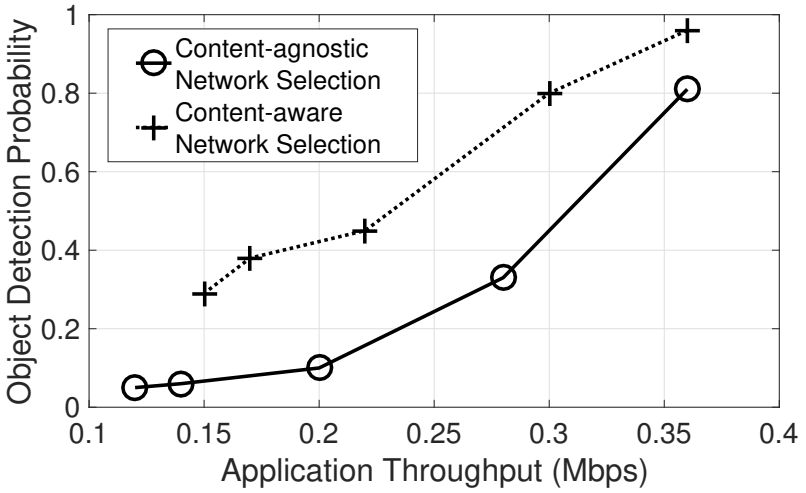


Figure 5.5: Object detection probability as a function of application throughput

5.5.2 Numerical Results

We performed the experiment with our testbed in a scenario where the WiFi network is affected by bursts of traffic competing with the video stream. We used controlled UDP flooding with the iperf [189] tool to produce the congestion in the WiFi channel. We generated the external traffic over WiFi on the same channel with packet bursts whose arrival is distributed according to an exponential distribution of fixed rate, and the burst size is 3

seconds. Bursts congest the network and result in localized sequences of packet drops.

From our preliminary study, we built a map between the QoC level and the packet loss rate based on which the edge processor sets the maximum packet loss threshold. In the high interference regime we used in our testing, the WiFi channel drops about 25% packets on average due to interference. Hence, in order to reduce the packet loss and fulfill the QoC requirements, the network selection may need to re-direct part of the stream to LTE in periods where the WiFi is affected by congestion. The frequency of offloading traffic to LTE depends on the threshold value set by the edge processor, which corresponds to the QoC requirement.

We performed the experiments by varying the QoC threshold and measured the object detection and throughput both for content-aware and content agnostic algorithms. Fig. 5.4 shows that when WiFi is used, in the considered interference regime object detection probability is close to zero due to almost one fourth of packet lost due to congestion. It can be observed that if the whole stream is assigned to LTE, then object detection probability is close to 1. When network selection algorithm is employed, the lower the QoC threshold, i.e., lower packet loss tolerance, the higher the object detection probability. The figure also shows that when content-aware network selection is employed, i.e. always using LTE to transfer reference frame and WiFi to LTE offloading for other packets when WiFi has average packet loss smaller than the threshold, then the overall performance of the system improves with respect to the content-agnostic network selection. However, the content-aware network selection may result in larger usage of LTE resources, and requires DPI.

In addition to QoC metrics, we measure the output throughput and object detection for both content-aware and content-agnostic selection. Fig. 5.5 shows that as the throughput increases, object detection probability increases as expected. It also shows that content-based network selection leads to greater improvement in terms of object detection with respect to throughput improvement compared to the content-agnostic case. We then measure the cost of

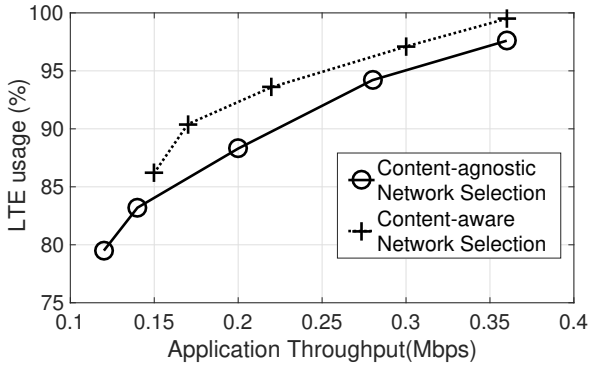


Figure 5.6: LTE usage in terms of network operation cost as a function of application throughput.

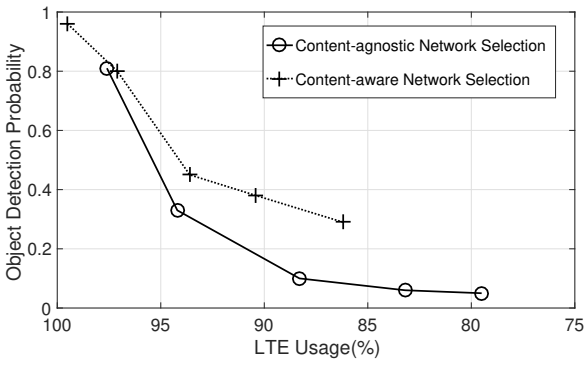


Figure 5.7: Object Detection Probability as a function of network operation cost in terms of LTE usage.

operation in terms of LTE usage for the same packet loss threshold range, and plot it against the application throughput in Fig. 5.6. It can be observed that throughput improvement comes at the price of an increased cost of network operation. In the considered setup, the cost is larger for content-based network selection than in the content-agnostic network selection mechanism.

Fig. 5.7 depicts object detection probability as a function of network operation cost in terms of LTE usage. It can be observed as the network cost decreases, object detection probability also decreases in both content-based and content-agnostic cases. It can also be observed that when the cost is very high ($> 90\%$), i.e., a small packet loss tolerance is used to trigger an offload to LTE, the object detection probability is high and comparable in the two cases. However, as the packet loss threshold is small, for a given cost of operation the content-aware network selection obtains larger object detection probability compared to the content-agnostic case.

Finally, we measure the average application latency and jitter. Fig. 5.8 shows that the average latency increases sharply when the throughput decreases, i.e. when the packet loss threshold is large. It also shows that when the achievable throughput is close to its maximum – that is, 0.35 Mbps – content-aware and content-agnostic selection result in similar latency.

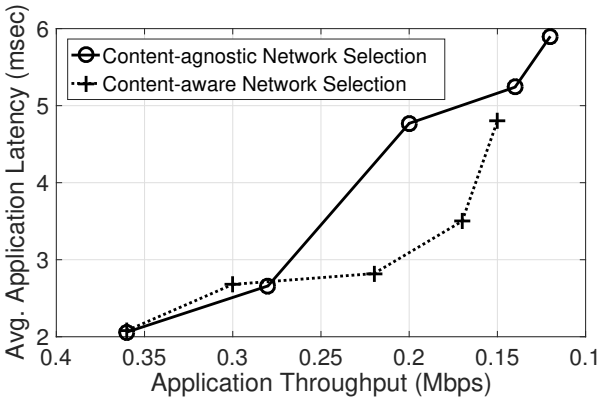


Figure 5.8: Application latency as a function of application throughput

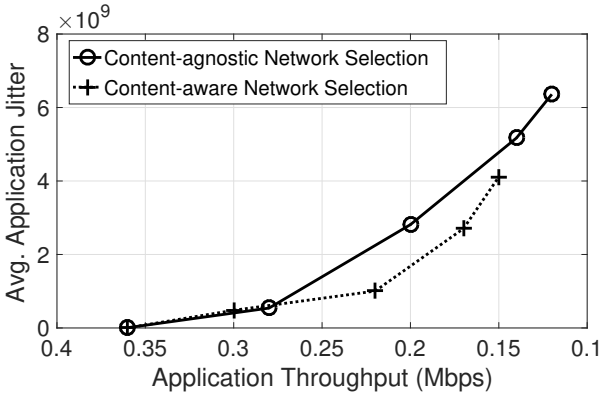


Figure 5.9: Application jitter as a function of application throughput

Thus, even if suffering a decrease in throughput, the content-aware protocol suffers a smaller penalty in terms of latency. Similar considerations can be made for application jitter as shown in Fig. 5.9.

5.6 Conclusions

The primary contribution of this work is a novel edge-assisted network selection mechanism for real-time services in urban IoT. We introduce the notion of Quality of Computing, where the edge processor, and the computational task, inform network selection. The network selection protocol differentiates control decisions based on content to further improve performance over cost of network operation given QoC requirements. An experimental evaluation is performed by transmitting a real surveillance video on a testbed built with open-source based WiFi and LTE networks. Numerical results show that in periods where the WiFi network suffers congestion and is unable to provide the desired QoC, the data can be effectively offloaded to LTE in real time to support computation.

Chapter 6

Data-driven Dynamic Path Selection for Real-time Video Applications

6.1 Introduction

Recently introduced paradigms where machines collaborate to accomplish a common task are imposing increasingly stringent constraints on the Quality of Service (QoS) achieved by the underlying communication layers. An illustrious example of such model, which have been receiving considerable attention, is edge computing [52, 177, 33], where a compute-capable machine positioned at the network edge provides data processing services to mobile devices interconnected through a low-latency, one hop, wireless link.

However, the volatile nature of wireless links may induce periods of time where the ability of mobile devices to deliver data to edge servers degrades. For instance, bursts of data transmissions from other wireless terminals active on the same frequency channel, or temporary loss of line of sight connectivity to the access point or base station in urban environments may drastically reduce achievable data transfer rate and result in large delays or packet loss.

Intuitively, these issues are particularly relevant when edge computing is used to support real-time mission critical applications.

The overall objective of this research is that of maximizing the integrity of data streams emitted from mobile devices and directed to the network edge under stringent capture-to-delivery delay constraints imposed by real-time applications. We specifically focus on one of the most challenging and relevant applications: real-time video streaming for immediate edge-assisted analysis. We consider a scenario where a mobile device and an edge server can communicate over multiple wireless channels – for instance, multiple Wi-Fi channels associated with one or more access points, and make the following assumptions: (*i*) due to mobility and activity from other active devices the characteristics – *e.g.*, short term throughput and packet-wise delay – of each channel present complex dynamics at different time scales; and (*ii*) devices can acquire information on the channel state – abstracted here as a *transformation* imposed to the packet stream – only by transmitting packets over available channels and observe their outcome. In this challenging environment, we formulate a channel selection problem whose goal is twofold: (*i*) maximize the perceived quality of the video stream at the edge server, while (*ii*) minimizing the impact of the information flow on the Quality of Service (QoS) of other active data streams. At the core of the problem is reliable channel *prediction*, that is, the ability of the channel selection logic to forecast future performance of the video stream to inform selection decisions.

Critically, as only the channel currently being used is *observed*, the selection logic is clueless about the current state of the other channels. In order to acquire complete state information, all the channels would need to be used simultaneously, thus considerably increasing network load, possibly degrading the QoS of other data streams. To solve this impasse, we propose a *probe-based* approach, where prediction over unused channels is informed by the transformation applied by the channel to a train of short packets periodically transmitted.

In summary, this research makes the following contributions:

(a) By means of detailed simulation, we create a comprehensive datasets based on real-world application layer traffic traces. The dataset connects features of packet delivery patterns – both from the video stream and probes – over multiple channels to the future perceived Peak Signal to Noise Ratio (PSNR) [106] of the delivered video. The environment captures complex traffic, and thus contention, patterns as well as mobility of the wireless nodes.

(b) We characterize the information on the future PSNR of the video stream contained in features of the packet streams – both video and probes.

(c) We develop a prediction framework based on state of the art classification algorithms, including Convolutional Neural Networks (CNNs).

(d) Based on the predictor's output, we implement an algorithm dynamically selecting the channel which most likely will provide the highest PSNRs in a predefined temporal window.

Results obtained by extensive simulations show that the prediction-based framework achieves an overall video quality close to that achieved by replicating the video stream over all the available channels while imposing a minimal increase in the degradation of other data streams compared to transmission over a single channel.

6.2 Preliminaries

From a high-level perspective, the development of a predictive framework to dynamically select channels providing the desired quality to a data stream in face of complex underlying traffic and channel gain presents two main technical challenges:

(ii) Develop an effective methodology to acquire information on the state of all available channels; and

(ii) Build effective strategies mapping the acquired state information onto future QoS or application layer performance of the data stream.

From the point of view of prediction, one of the main sources of complexity is the possibly fast rate of change of channel conditions induced by the complex traffic patterns generated by other active data streams, their activation/deactivation patterns, as well as the convoluted interactions between data streams due to control protocols at the different layers. This aggregates with variations in the channel gain and physical parameters due to mobility and micro and macro features of the environment. The result is a nested set of – inherently stochastic – processes controlling the *transformation* applied by the network to packets transmitted over the channel.

Intuitively, the characterization and even representation of these processes is extremely complex. Furthermore, most of the involved variables (*e.g.*, inner state of channel access or transport layer protocols) are not observable. We, then, take a data-driven approach, and study the information contained in the transformation of packet trains itself on the future of the channel state, here intended as the PSNR of the video if a specific channel were to be selected. We address the problem of acquiring information on channels that are not currently used for video transmission, by introducing in this context the notion of probing through lightweight packet bursts.

Fig. 6.1 shows the general diagram of the proposed framework: features extracted from the video and probe streams are used to predict the future quality of the video if any of the C channels were to be used for transmission. The individual components of the framework and the rationale behind its design are explained in the following.

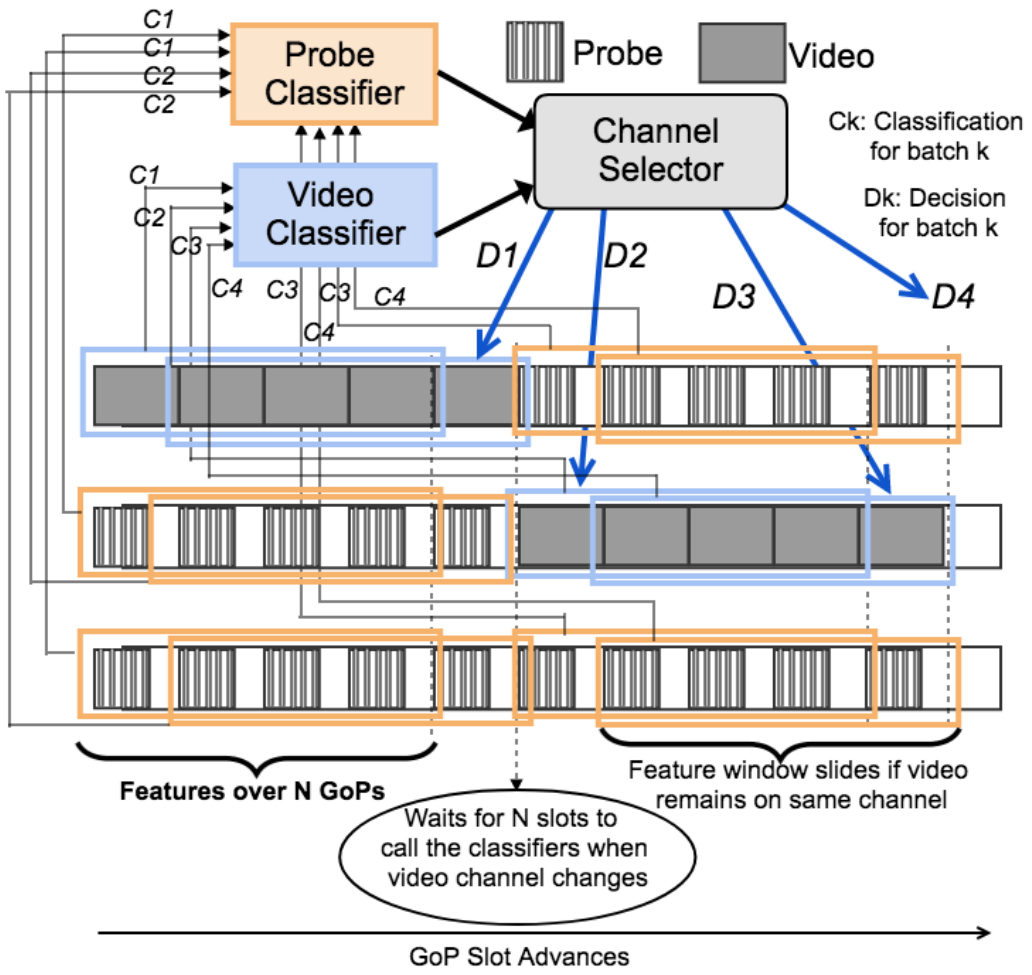


Figure 6.1: High level schematics of the channel selection framework.

6.2.1 Scenario

We consider a scenario in which a video stream captured in real-time by a mobile user needs to be transported to an edge server. The user and the edge server can communicate through C independent wireless channels – for instance corresponding to different access points in the vicinity – indexed with c=1, . . . , C. The channels are shared with other users and data streams, which all contend for the available frequency resource.

The video stream is encoded, packetized using UDP, and transmitted over one of the wireless links to the edge server. The frame rate of the video is F_r frames/s, and we assume that all the packets referring to frame i, i=0, 1, 2, . . . are injected into the transport layer socket at time

i/F_r. Temporal encoding is used to reduce the bandwidth used by the video, where frames are organized into Group of Pictures (GoP) composed of a reference frame and a series of differential frames encoding the differences with respect to the corresponding reference frame. Due to the real-time nature of the application, we assume differential frames only depends on the previous reference frame, and that GoPs are composed of a fixed number of frames F. We impose a strict deadline from the capture of a frame and its delivery. We denote the maximum capture-to-delivery time with Δ. Packets delivered after the deadline are discarded, and the decoder will only use the received ones.

6.2.2 Selection Problem

Due to practical considerations, we define the selection process at the temporal granularity of GoPs. Let's index GoPs with p=1, 2, ... We, then, denote the average Peak Signal-to-Noise Ratio (PSNR) of GoP p with $\Phi(p)$, computed as the average PSNR of all the component frames, and define the control variable $c(p)\in(1,\ldots,C)$ corresponding to the channel selected for the transmission of GoP p.

In order to maximize the quality of the video, we simply aim at the selection of the channel providing the maximum average PSNR, that is,

$$c(p)=\arg\max_{c}\Phi(p,c). \tag{6.1}$$

Rather clearly, the selector needs to make the decision before the GoP is transmitted, and selection is necessarily based on prediction.

6.2.3 Prediction

We define prediction as a classification problem, where classes are defined based on functionals of the GoPs' PSNR within a window of future GoPs. This is motivated by the inherent difficulty in the short-term prediction of individual GoPs' PSNR values in network environments characterized by mobility and complex traffic patterns. Let's set the current GoP index to p=0, we then define Z classes $z\in(1,\ldots,Z)$ as

$$z = \mathcal{F}_W(\mathbf{\Phi}_{1,W}), \tag{6.2}$$

where $\mathcal{F}_W$ is a functional mapping of the vector $\mathbf{\Phi}_{1,W}=(\Phi(1),\ldots,\Phi(W))$ to the class z.

Several choices of the classification function $\mathcal{F}_W$ are possible, including a simple discretized average over the PSNRs within the temporal window. Herein, we choose a function which aims at preserving a general good quality of most GoPs in the window. Specifically, we define

$$\mathcal{F}_W(\mathbf{\Phi}_{1,W})=\begin{cases} 1 & \text{if} \quad \sum_{p=1}^{W} I(\mathbf{\Phi}(p) > y)\geq k, \\ 0 & \text{otherwise,} \end{cases}$$

where y is a predefined PSNR threshold, $k\in\{0,\ldots,W\}$ and $I(\cdot)$ is the indicator function. Thus, we say that the channel in the window is "*good*" (z=1) if at least k GoPs have a PSNR larger than y. The selector, then, will attempt to assign the stream to a *good* channel.

We now define the input of the predictor. As mentioned earlier, due to both technological and environmental parameters and their variations, the channel applies a transformation to the packet transmitted by the mobile user. Define $\mathbf{\Omega}(p) = \begin{bmatrix} \boldsymbol{\tau}(p) \\ \boldsymbol{\lambda(p)} \end{bmatrix}$ as the matrix composed of the row vectors $\boldsymbol{\tau}(p)$ and $\boldsymbol{\lambda}(p)$, whose elements are the injection time and size of all the

packets transmitted during GoP p. We characterize such transformation using the function $\mathcal{T}(\cdot)$ as follows

$$\begin{bmatrix} \tilde{\boldsymbol{\tau}}(p) \\ \tilde{\boldsymbol{\gamma}}(p) \end{bmatrix} = \mathcal{T}(\boldsymbol{\Omega}(p), \mathbf{s}(p)), \tag{6.3}$$

which takes as input $\boldsymbol{\Omega}(p)$ and a generic vector of latent variables $\mathbf{s}(p)$ to return the vectors $\tilde{\boldsymbol{\tau}}(p)$ and $\tilde{\boldsymbol{\gamma}}(p)$, which contain the delivery time and a delivery flag (equal to 1 if the packet is delivered by the deadline and 0 otherwise), respectively. Our objective is not that of modeling the, extremely complex, function $\mathcal{T}(\cdot)$ or define the state vector $\mathbf{s}(p)$. Instead, we aim at characterizing and using the information contained in that transformation on the future performance of the video stream over a channel.

We, then, define the predictor $\overline{z} = \mathcal{P}(\boldsymbol{\Pi})$, where $\overline{z} \in \{0, 1\}$ is the predicted class and $\boldsymbol{\Pi}$ is a matrix

$$\boldsymbol{\Pi} = \begin{bmatrix} f_1(0) & f_1(-1) & \dots & f_1(-H) \\ \vdots & \vdots & \dots & \vdots \\ f_F(0) & f_F(-1) & \dots & f_F(-H) \end{bmatrix} \tag{6.4}$$

whose rows correspond to different features and columns to different GoP slots. The features are extracted from the matrix

$$\begin{bmatrix} \tilde{\boldsymbol{\tau}}(0) & \tilde{\boldsymbol{\tau}}(-1) & \dots & \tilde{\boldsymbol{\tau}}(-H) \\ \tilde{\boldsymbol{\gamma}}(0) & \tilde{\boldsymbol{\gamma}}(-1) & \dots & \tilde{\boldsymbol{\gamma}}(-H) \end{bmatrix}. \tag{6.5}$$

We note that the features needs to be calculated at the access point or edge server, which feeds them back to the user by means of a short packet. In order to ensure a reliable delivery of this packet, features associated with the last frame of the last GoP may be omitted.

6.2.4 Probing

The input of the predictor are features computed from the outcome of packets transmitted over the channel in the T preceding temporal slots (corresponding to GoPs). Clearly, computing such features is possible only if the channel currently used to transmit the video. To overcome this issue, we introduce and study in this scenario the notion of *probes*. The core idea is to acquire information by periodically transmitting short trains of packets. This reduces the impact on other active streams compared to transmitting sections of the video. As for the video stream, information acquired by the probes takes the form of features associated with the transformation $\mathcal{T}(\cdot)$.

Intuitively, the *shape* of the probe influences the amount of information harvested when observing its outcome. Herein, we synchronize probe transmission with the starting of GoPs, and compose the probes of a train of Q packets of size S transmitted in a single burst. We test different sizes and number of packets to evaluate their ability to predict future video quality with respect to their impact on the data streams populating the unused, probed, channels. The results are reported and discussed in Section 7.5.

6.3 Network Environment and Dataset

This section introduces the environment created to collect feature matrices from the video stream and probes, as well as to test the channel selection technique. Our objective is to capture the complex interactions between the video stream and the traffic patterns associated with relevant applications. We focus on the Wi-Fi communication technology, which has a high degree of unpredictability due to the contention mechanism and the low amount of control imposed by the infrastructure, and associate each of the C channels to an access point. The investigation of a multi-technology environment, for instance including connection to

LTE eNodeBs is left to future studies.

Importantly, available network datasets are mostly at the transmission level, that is, they capture the timing of packet transit *after* the control imposed by all the network layers. We contend that a key feature to build effective and realistic predictors is to capture the intricate interactions between transmission protocols given the dynamics of the traffic injection and environment. Then, in addition to the user streaming the video, the environment is populated with M mobile users whose association is equally divided across channels. We characterize each node m by defining a "class" $c_m \in \mathcal{C}$, where each element in $\mathcal{C}$ corresponds to a description of the application level traffic generated by the node. Although a model-based approach is possibly viable, we take a data-driven approach, where each class corresponds to traces extracted from real-world applications. We include the following applications: (*1*) Youtube video streaming, (*2*) Skype voice call, (*3*) File Transfer Protocol, (*4*) Skype video call and (*5*) collection of background web traffic that persistently inject traffic in the network. Importantly, the traces we collect are at the application layer, that is, they are composed of a sequence of packet injection times and sizes, where the time corresponds to the injection of the packet in the transport-layer buffer. A high-fidelity representation of the interaction between nodes is achieved by using NS-3 to simulate transport layer and link layer mechanisms.

Although certainly not exhaustive, the considered set of applications includes representatives from a broad spectrum of typologies, ranging from large burst of traffic (FTP), to a continuous, but variable, stream of packets (Youtube and Skype), and semi-deterministic traffic patterns (background web traffic). Figure 6.2 shows the traces collected for the mentioned classes of applications and combined complex structure of the overall envelope of the traffic shape when all of them are active.

In addition to the application-specific variations in the pattern of injected packets created by the traffic traces, we consider stochastic activation/deactivation patterns. In particular, we define a Poisson process with parameter β, where the events generated by the process

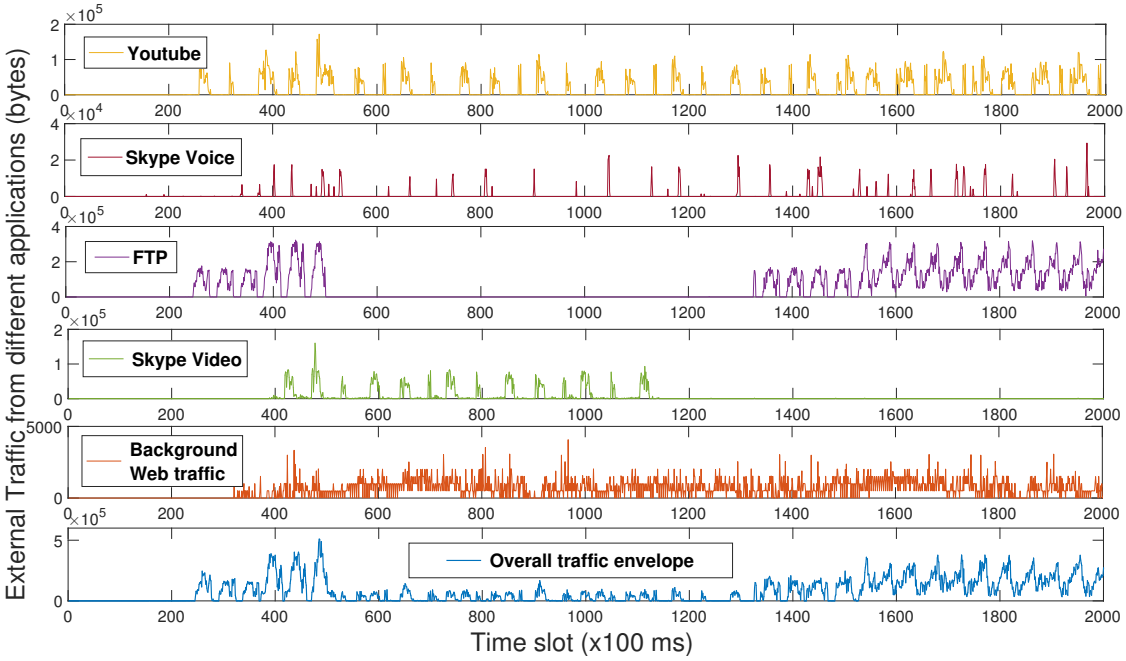


Figure 6.2: Envelope of the traffic classes at the network level. The set includes Youtube, Skype voice and video, and FTP applications as well as persistent background web traffic.

correspond to the activation/deactivation of a stream. Specifically, each event is marked with either 1 (probability ρ) or 0 (probability $1-\rho$). Event marked with 1 correspond to node activation, and unless all the nodes are already active one of the nodes is selected at random and activated. Conversely, if the event is marked with 0 and at least one node is active, then one of the active nodes is selected at random and deactivated.

The used video has a resolution of 1920x1080 and encoded with H.264/AVC [203] format with average encoding bitrate of $1Mbps$ using ffmpeg [190]. We generate a series of Transport Stream (TS) packets of fixed size 188 bytes and use UDP as transport layer. Several TS packets are encapsulated in UDP packet and transmitted over the network. All the other data streams are transported using TCP. We set N=3, and use the parameters specified in the table 6.1.

To extract the dataset, we create 3 non-overlapping channels in the NS-3 simulator, and run simulations for 50000 GoPs. Each GoP is composed of 30 frames transmitted over 1

Parameters	Value
WiFi Standard	IEEE 802.11g
WiFi Bandwidth	10 MHz
No. of WiFi APs	3
WiFi Bands for APs	Channel 1, 6, 11
Initial XY of Mobile node	(0,0) meters
Initial XY position of APs	(25,0), (0,45), (45,45) meters
Propagation Model	Cost231PropagationLossModel
WiFi Rate Adaptation	ARF Algorithm
Mobility of Mobile Node	RandomWaypointMobilityModel
XY bounds for Mobility	(0,0), (0,60), (60,60), (60,0)
Min Mobility Speed	0.9 m/s
Max Mobility Speed	5 m/s
Min Pause for Mobility	1 sec
Max Pause for Mobility	6 sec
GoP Delivery Deadline	1 sec

Table 6.1: Simulation Parameters.

sec. Each video frame of the GoP is transmitted as a burst of packets. Since the video is temporally encoded, the first frame of the GoP is a reference frame with larger size of 42 UDP packets of 1400 bytes, all following differential frames are smaller and contained in a burst of 2 to 4 UDP packets. Within any GoP slot, we transmit the video data on one of the 3 channels and a burst of probe packets at the beginning of the GoP on the the other channels (see Fig. 6.1). The channels have the same set of contending application streams and statistics of the activation/deactivation processes. Heterogeneity of channel conditions among the 3 channels are induced by setting a different initial offsets of the traffic injection traces as well as by means of the randomized activation/deactivation processes.

The result is a highly complex and varied temporal pattern of channel conditions over the channels. Fig. 9.4 depicts an example of temporal trace of the PSNR and average packet delay at the GoP granularity. Periods of high channel load with different characteristics can be observed, interluded by periods of milder traffic conditions. Note the non-trivial relationship between packet delay and PSNR due to the complex characteristics of video encoding.

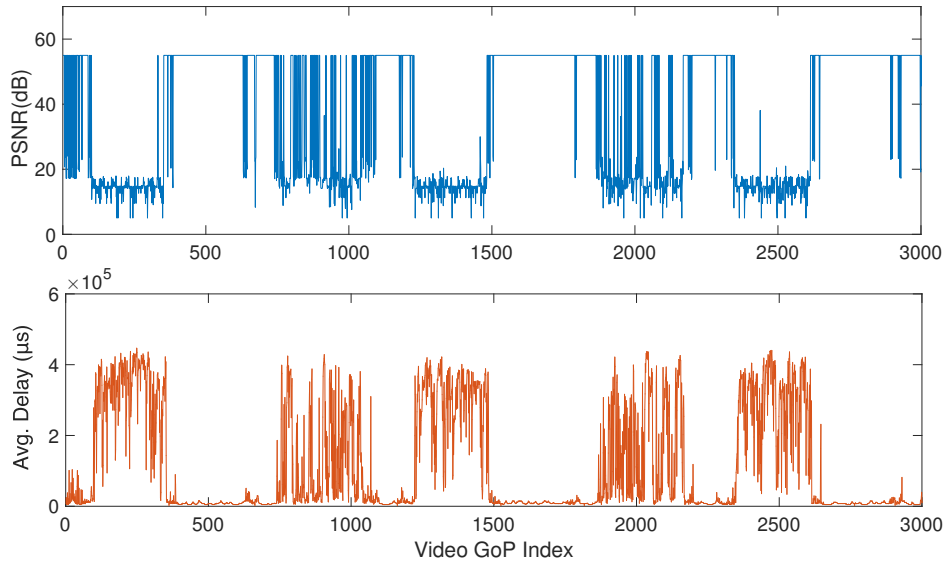


Figure 6.3: Example of temporal pattern of PSNR and average packet delay induced by the traffic traces and underlying activation/deactivation process.

To mimic the effect of selection, and collect data on the dependency between probe transformation and future video quality, the channel used to transport the video is randomly selected. The frequency of switching is changed across experiments, that are merged into two large databases containing video or probe features history and the class of the following GoPs. Specifically, we set the prediction and history window to have size 10 GoP slots, and use a sliding window of size 20 GoP slots. In the video to video database, the features are computed on the first 10 slots, and the class is computed on the following 10 slots . For the probe dataset, the data points are collected when channel switching happens and 10 consecutive GoPs are transmitted after 10 probe bursts.

6.4 Prediction Framework

The dataset described in the previous section is used to build the predictor $\mathcal{F}$. Note that two separate prediction functions are used for probe-based and video-based prediction. First, we describe the extracted features and analyze their prediction power. Then, we describe the

structure used to define the predictor.

6.4.1 Feature Analysis

Notably, PSNR, a widely used feature in video quality prediction, is available only if video frames are transmitted. We define, then, a set of features that are available both when the video or the probes are sent. Specifically, we compute: $f_1(p)$ - average delay; $f_2(p)$ - delay variance; and $f_3(p)$ and packet loss rate associated with packets transmitted in the time interval corresponding to GoP p. Note that packet loss may be also because of application layer discarding due to delivery delay exceeding the deadline.

We analyze the predictive power of the features across the temporal window $-H, \ldots, 0$, which we refer to as *the history*. To define the classes z=1 and 0 – high and low video quality, respectively, We set the PSNR threshold to 40, and the number of above-threshold GoPs to k=6. As in the dataset collection, we set the prediction and history window size to 10. Unless otherwise specified, in all plots probe bursts are composed of 50 packets of size 100 bytes.

Fig. 6.4 shows the separability of each class based on the last GoP slot (index 0). It is apparent how each feature contributes to the overall – imperfect – separability between the two classes. We formalize the prediction power using the mutual information, a metric that has been widely used for feature selection, between the features and the class variables. Given two random variables x and y, their mutual information is defined from their probability density functions $p(x)$, $p(y)$, and $p(x, y)$ as:

$$I(x; y) = \iint p(x, y) \log \frac{p(x, y)}{p(x)p(y)} \, dx \, dy. \tag{6.6}$$

Figures 6.5a and 6.5b show the mutual information between each feature and the class labels

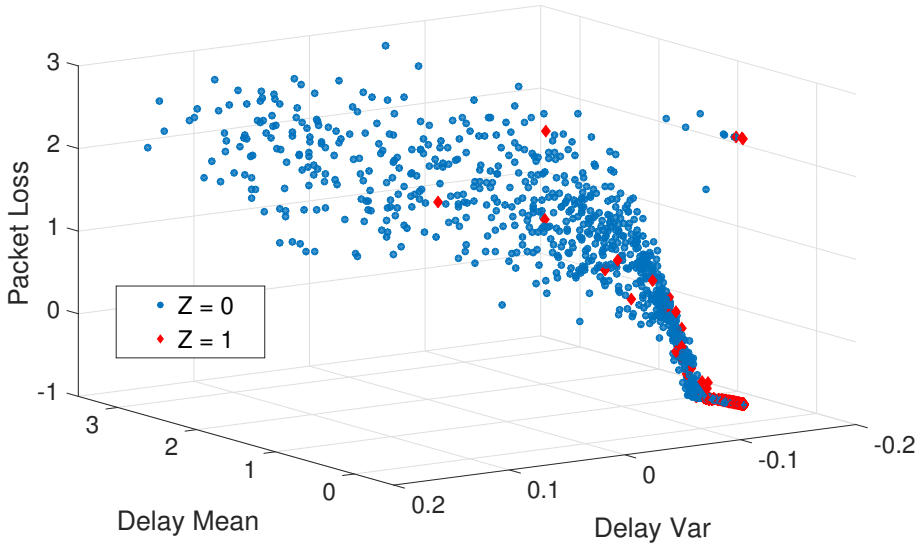
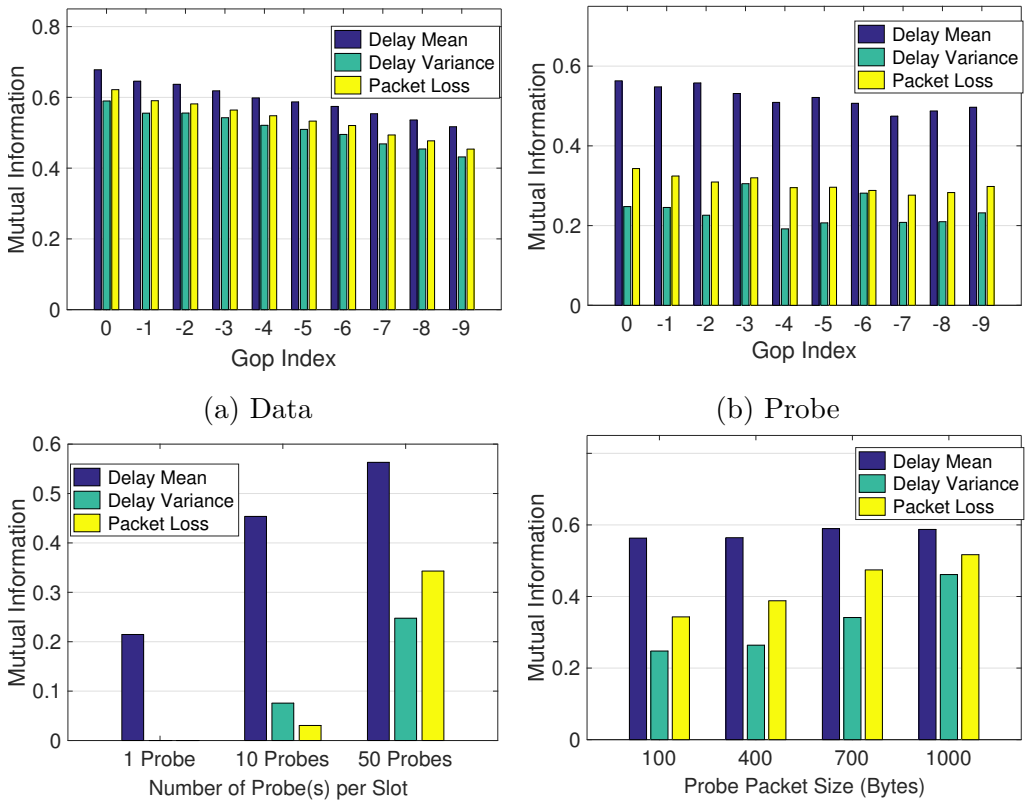


Figure 6.4: 3D visualization of the distribution of 3 feature types for the two class labels, where z=1 and z=0 correspond to a *good* and *bad* channel, respectively.

over a history of 10 slots for video and probe's packets, respectively. First, we consider a case where nodes have fixed positions. It can be observed how features from past GoP slots still present a non-negligible correlation with the future class label. Quite interestingly, while the mutual information computed on features on the video packet stream has the expected decreasing trend as the temporal shift increases, the trend associated with the probes presents oscillations and heavily penalizes delay variance and packet loss rate. The former effect suggests that past features can carry information on past events, which may have long term significance and may impact the class label due to transport and network-layer protocols. The latter emphasizes the weaker ability of short bursts packets to collect information along some feature dimensions.

Fig. 6.5c shows the mutual information of the different features of the last GoP slot as a function of the number and size of the packets contained in the probes. A perceivable increase of all the features as the probe number is increased can be observed. This is due to the fact that a larger number of packets allow the probe to sample beyond local conditions of the network, as well as a more accurate estimation of the features. An increasing information is



(a) Data (b) Probe

(c) Number of probes (d) Probe size

Figure 6.5: Mutual information between the features in different GoP slots and the class label for video (a) and probe (b) packet streams. Plots (c) and (d) show the trend as a function of packet number and packet size of the probe bursts, respectively. The most recent GoP slot is used in the latter two plots.

also observed as we increase the size of the probe packets to match that of the packets from the video stream (Fig. 6.5d). This is related to the different effect of network parameters on smaller packets, which intuitively have a reduced interaction with other data streams. However, longer bursts of bigger packets also induce a larger degradation to coexisting data streams.

Finally, we evaluate the impact of nodes' mobility on the predictive power of the features. Fig. 6.6 shows the ratio between the mutual information – across features and slots – associated with moving (see mobility parameters in Table 6.1) and stationary nodes. Interestingly, while a general degradation affects the mutual information, some features/slot pairs gain predictive power, suggesting some long-term correlation present in the path loss induced by

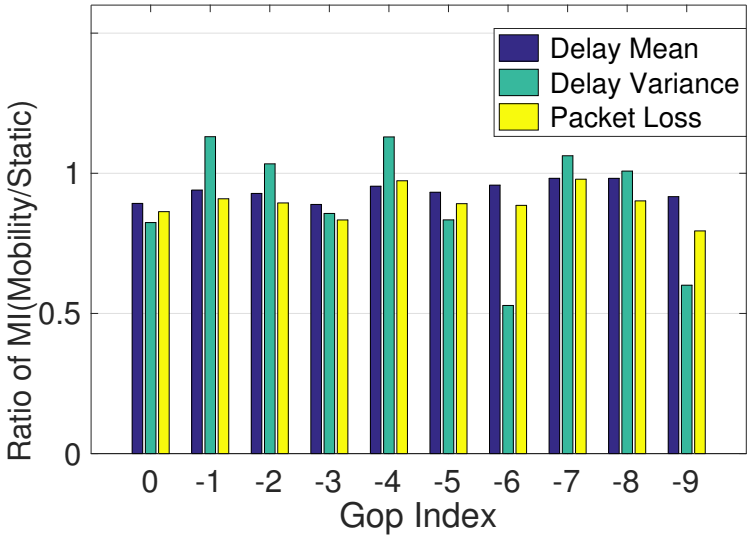


Figure 6.6: Ratio between the Mutual Information computed in the mobility and static mobility cases (mobile/static).

the mobility model.

6.4.2 Classifier

Based on these features, we now build the predictor $\mathcal{F}$ using state of the art machine learning models. Clearly, we need to define two distinct classifiers, translating video stream and probe stream features into the binary class we use to represent the quality of future video GoPs.

In order to fully exploit the temporal dependencies which exist between the features and the future class, we use a deep Convolutional Neural Network (CNN) [126]. This choice also allows the use of the CNN soft labels' output to improve selection accuracy, as explained later. A CNN is comprised of one or more convolution layers – often with a subsampling step – followed by one or more fully connected layers as in a standard multi-layer neural network. The architecture of CNNs is designed to take advantage of the multi-dimensional structure of the input. This is achieved using local connections and weights followed by pooling, which results in translation invariant features. Importantly, CNNs are easier to train and have very

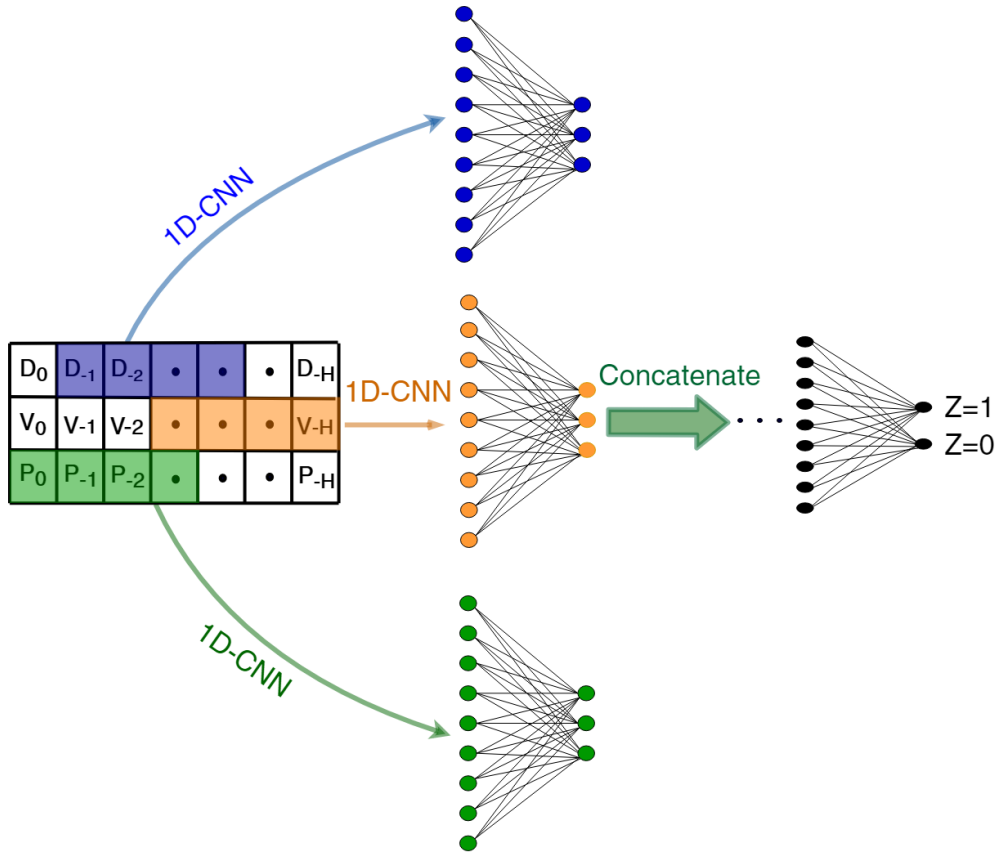


Figure 6.7: CNN Architecture used to build the predictor. The model takes as input a vector with shape $3 \times (H+1)$, in which each row is associated with one feature type D, V and P to indicate average delay, delay variance and packet loss, respectively. Then a 1D CNN is applied to each row to analyze the temporal fluctuations and trends of each feature type, and a the filtered data is unrolled and fed to a fully connected neural network to extract the output class label.

fewer parameters than fully connected networks with a comparable number of hidden units.

In the considered case, we use the feature matrix $\mathbf{\Pi}$ as a 2-D input set feature. Then, Λ different 1-D kernels (filters) are applied on the each row of the input matrix. Assuming that the size of all kernels is the same, and is equal to T_k which $T_k < H$. Kernel K_λ can be described as a vector as below:

$$K_\lambda = \begin{bmatrix} \kappa_0^\lambda & \kappa_2^\lambda & \dots & \kappa_{T_k-1}^\lambda \end{bmatrix}$$

The output of the convolution between K_λ and input feature vector is a vector with length

$L = H - T_k + 1$ in which the element $R^i_\lambda(l)$ is obtained as follows:

$$R^i_\lambda(l) = \sum_{p=0}^{T_k-1} \kappa^\lambda_p f_i(l-p) \tag{6.7}$$

Where f_i in the equation above can be any of three feature f_1, f_2 or f_3. After convolving with the features, each kernel roughly summarizes key trends in the feature progression. For example, a kernel can detect whether there is a decreasing or increasing trend in recent slots, or capture a peak or valley within the window. A larger number of kernels allows the model to capture a broader range of fluctuations and trends in the features' history.

After that, all $R^i_\lambda(l)$ are unrolled and passed through a non linear activation function, which is, then, fed to a deep fully connected neural network as shown in Fig. 6.7 to output the class labels.

6.4.3 Channel Selection

The goal of selection decisions is to maximize the short-term future PSNR of the video given the current state of all the channels. Due to prediction errors, a channel whose predicted label is z=1 may still present a larger than expected number of low PSNR GoPs. To mitigate the impact of prediction errors, we *rank* the channels using the soft output of the corresponding predictors and select the channel with the highest rank. Therefore, the policy for the channel selector is to choose a c^* channel that

$$c^* = \arg\max_c \hat{z}_c, \tag{6.8}$$

where $\hat{z}_c$ is the soft output of the classifier in channel c.

6.5 Performance Evaluation

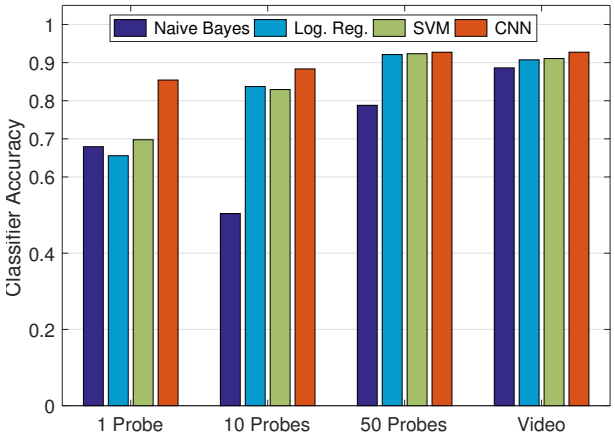
We provide here an extensive evaluation of the offline and online performance of classification and channel selection. We used PyTorch [118] to build the CNN model, and the trained model is integrated into the NS-3 network environment for online performance test. To build the CNN model, we use $\Lambda = 10$ different kernels and set the kernel length to $T_k = 5$ which is equal to half of the history size. We set the learning rate to $l_r = 0.003$ and use the Adam optimizer [120] to train the model. We train and test the classifier on the probe and video datasets. Each dataset includes approximately 10000 video and 2000 probe datapoints. We use %75 of datapoints for training and the remaining %25 to assess offline classification performance.

6.5.1 Offline Classifier Performance

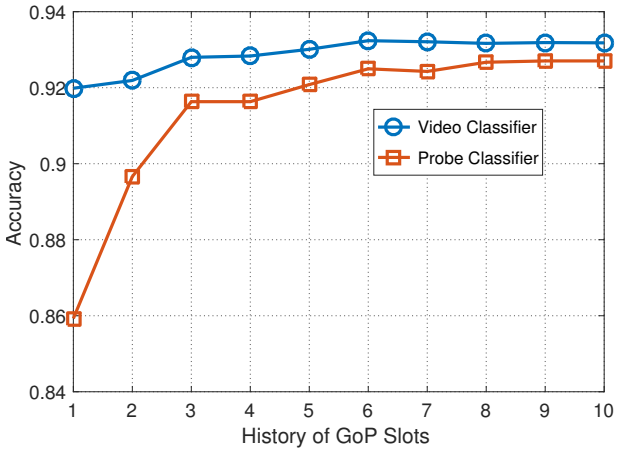
We first report micro-benchmarks on the performance of the classifier model in terms of prediction accuracy.

Figure 6.8a compares the classification accuracy achieved by probe composed of a different number of packets (packet size equal to 100 bytes) and that achieved based on the video stream. Interestingly, the large amount of information collected by past video packets – which are larger in size and number compared to probes – allows accurate classification even when using simple classifiers. Conversely, accurate classification based on lightweight probes necessitates complex classifiers, capable of separating the two classes using more convoluted surfaces. This motivated our choice of CNN as the core of the proposed channel selection framework.

We analyze the effect of the history size H on the classification performance in Fig. 6.8b. Interestingly, while the classifier based on the video packet stream suffers a relatively small



(a) Performance Comparison of different classifiers for different batch sizes of 100 bytes probe and the video stream-based classifier.



(b) Comparison of the effect of the history length on classification accuracy..

Figure 6.8

accuracy loss when the history size is reduced, the probe-based classifier necessitates a larger amount of GoP slots to provide comparable performance. This is connected to the need of a larger set of packets spread over time to adequately sample the network state and its dynamics.

6.5.2 Online Channel Selection

First, in order to fully motivate the use of probes, we measure the degradation – in percentage with respect to the case with no video or probe stream – to other data streams' throughput caused by different probe models and the video stream itself (Fig. 6.9). We can observe that an already effective set of probes (burst size equal to 10) causes a 1% throughput loss, compared to the 43% caused by the video stream.

We now analyze the performance of the data-driven channel selection framework in terms of output video quality. We compare the data-driven selector with the following cases:

(i) *Fixed Channel:* The entire video is sent over one channel.

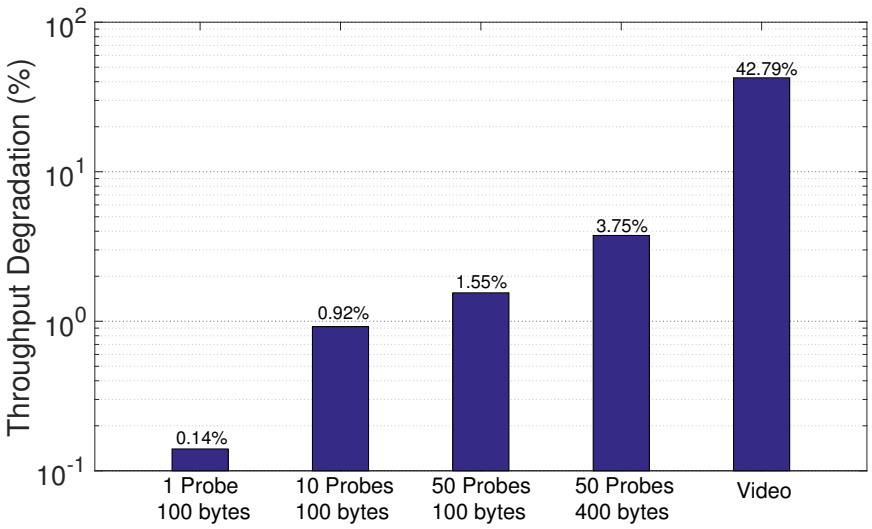


Figure 6.9: Throughput degradation of other applications imposed by probes or video streams.

Avg. Encoding Bitrate	FileSize (Bytes)	Avg. PSNR (dB)
800 Kbps	104977	36.95
600 Kbps	78772	35.82
400 Kbps	53241	35.01
200 Kbps	27486	30.03
100 Kbps	16056	26.65
50 Kbps	13912	24.47
25 Kbps	10865	18.63

Table 6.2: Mapping of bitrate to file size and achievable PSNR for ABR based streaming of the selected video.

(ii) *Adaptive Bit Rate (ABR) with full knowledge:* Assuming full knowledge of the channel throughput, we implement an ABR framework matching the available throughput to a coding rate. In order to compute the PSNR, we do not adapt the resolution of the video as in DASH [182]. We, then, pre-encode different versions of the video at different bitrates and use the map reported in table 6.2 for online adaptation. In realistic conditions, the adaptation of the encoding rate in real-time streams is rather difficult due to the high computation demand.

(iii) *Delay-Based Simple Channel Selector:* We implement a simple delay-based selector, which compares the average packet delay in the last GoP slot for all the 3 channels and

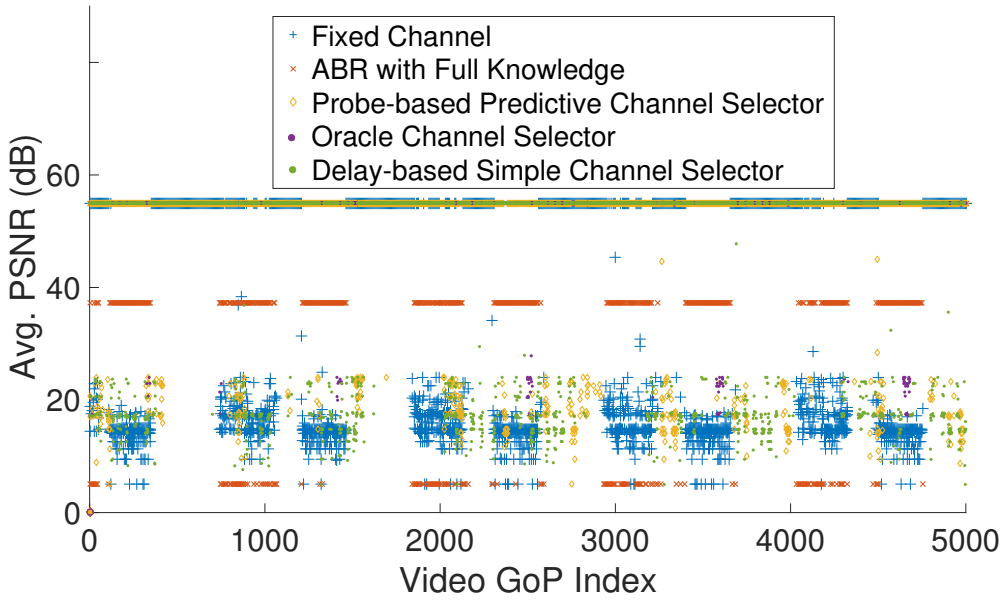


Figure 6.10: Temporal pattern of PSNR achieved by the different transmission strategies.

selects the channel with the smallest delay.

(iv) *Oracle Channel Selector:* As an absolute upper bound, we consider an oracle channel selector that has a priori knowledge of the PSNR in all available channels.

Fig. 6.10 shows examples of temporal patterns associated with the different selectors. Note that perfect PSNR corresponds to infinity. We then set the maximum PSNR value at 55 dB (similar to that achieved when 1 packet per frame is lost) for illustration purposes and to compute averages and distributions. It can be observed that the fixed channel case has several clusters of low PSNR most likely due to the simultaneous activation of many streams. By adapting the bitrate to the throughput, the genie-aided ABR mitigates this effect in some portions of the trace. The proposed probe-based predictive channel selection opportunistically re-routes the video through channels capable of supporting the video with full quality.

The average PSNR achieved by the different selectors in scenarios where nodes are static or

PSNR (dB)	**Fixed Channel**	**ABR with full knowledge**	**Oracle Channel Selector**	**Delay-based Simple selector**	**Probe-based predictive selector**
No Mobility	37.52	45.27	54.31	47.90	51.85
With Mobility	35.24	42.54	49.13	45.31	48.33

Table 6.3: Average PSNR comparisons in different approaches of the video streaming

move according to a random walk is shown in Table 6.3. It can be seen how the proposed data-driven approach (51.85 dB) performs closely to the Oracle channel selector (54.31 dB), with a loss of less than 3 dB, and an improvement of 14 dB with respect to fixed channel transmission. ABR transmission and Delay-based selection, at 45.27 and 47.90 dB, achieve intermediate performance. When considering high mobility, we note a reduction in the achievable average PSNR. However, also in this case the proposed predictive channel selector outperforms other options and performs close to the oracle selector.

The CDF of the PSNR is shown in in Figure 6.11. Transmission over a fixed channel results into a widely spread distribution of PSNR, which may impair the ability of the edge server to perform analysis. The probe-based classifier is capable of eliminating most low and mid-range PSNRs, although prediction errors result in a larger probability of mid-range PSNR than that achieved by the oracle. ABR transmission efficiently removes most low-range PSNRs, but the CDF has a sharp increase at 40 dBs, emphasizing the limits of adaptation against bad channel conditions, where the compression necessary to effectively deliver the packets produces a perceivable degradation.

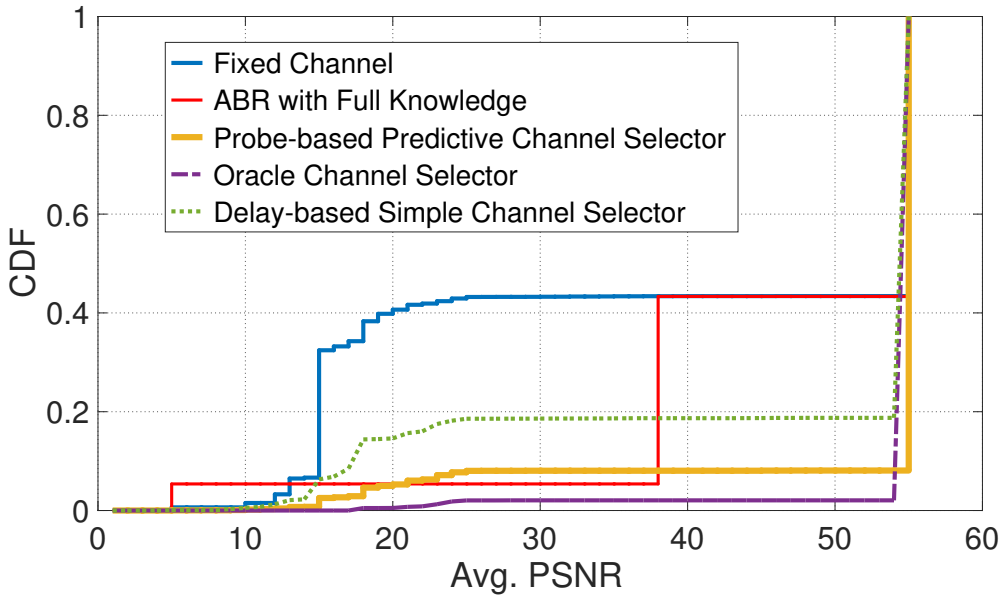


Figure 6.11: CDF of PSNR for the different transmission strategies.

6.6 Related Work

Adaptive Video Encoding: The majority of the adaptive video streaming techniques focus on the efficient and resilient delivery of stored video content. Dynamic Adaptive Streaming over HTTP (DASH) [182] and similar protocols [122, 162, 212] were developed to adapt the quality of the transmitted video – that is, its resolution, frame rate, etc. – to the capacity of the channel. The stored video content is encoded with Advanced Video Coding (AVC), where the frame sequence is spatially and temporally encoded, and multiple versions of each video segment are created with different bitrate. The objective is to provide the best Quality of Experience (QoE) to users by maximizing the information transported in the stream – the bitrate – while controlling the reliability of its delivery over unstable links to the final destination. An example within this class of approaches is Scalable Video Coding (SVC) [172], which provides a flexible coding scheme to support adaptive video transmission.

Clearly, adaptive video transmission relies on some form of estimation of the channel quality. Several contributions address this aspect of the problem [68], and propose frameworks mostly

based on the recent history of metrics such as PSNR or application variables such as playout buffer size. Importantly, these techniques rely on the transmission of the video stream for the extraction of observations on channel quality.

Recent work focuses on wireless networks, e.g., PiStream [207], and directly use physical layer information to control the video stream. However, this approach has limited applications to scenarios where congestion and traffic dynamics play a significant role in determining the channel capacity, and where the network has limited control over resource allocation.

Streaming over Multipath: There is a relatively limited body of work studying multi-path techniques for video transmission. The main advantage of multi-path transmission, where the stream is divided into sub-streams transmitted over different channels, is the possibly higher degree of diversity and the ability to harness the capacity of multiple resources. However, multi-path introduces challenges in terms of packet reordeing and head-of-line blocking. Multipath TCP (MPTCP) [41] was developed to address these challenges in general data streams by using different flow control and congestion control mechanisms, along with overlay solutions [40] to mitigate the head-of-line block issue when the paths have strongly asymmetric conditions. Based on MPTCP, Multi-Path DASH (MPDASH) [96] has been proposed to extend DASH to multi-path scenarios. However, MPDASH is based on MPTCP, and thus comes with all the limitations of TCP in a real-time streaming scenarios. Additionally, TCP-based techniques are not a good match to wireless scenarios due to their inherent fast variability. Finally, these techniques presume the use of all the channels simultaneously, with a possibly long adaptation time when a new path is added or removed, an event which may frequently occur in mobile wireless networks.

Machine Learning based Channel Quality Prediction: A number of recent contributions use machine learning techniques to predict the state of wireless channels. For instance, [69] uses a machine learning algorithm to predict when a burst of traffic will start or end in a wireless network. The outcome is used to switch the device state from active to idle

and idle to active to reduce power consumption. The authors used a data-driven approach to estimate the interference between two nodes from a merged packet trace which was collected via distributed sniffing. A decision tree based classification algorithm is employed in [14] to choose the best available network between a 3G cellular and a WLAN using features such as RSSI, location and type of application. In [117], Support Vector Machines (SVM) is used to estimate the throughput and delay of an Access Point's user in a WiFi network, given the traffic volume and signal strength of active interferes as input features. A recent work presents "PenSeive" [144], an Adaptive Bit Rate (ABR) video transmission framework based on reinforcement learning and neural networks. The primary goal is to learn from past decision on output bit rate with respect to channel state. However, these prediction techniques are focused on the physical layer, with no consideration of channel use patterns.

6.7 Conclusions

This work presents a data-driven form of prediction and channel selection for wireless channels subject to interference and congestion. The core idea is to build classifiers capable of transforming features collected in real-time by wireless nodes into predicted quality of the data transmitted to edge servers. We specifically consider a scenario where a real-time video stream is acquired and transmitted over highly dynamical channels. An extensive discussion on the predictive power of features extracted from the data and probe stream was provided. We measured the performance of classification and channel selection by means of extensive simulations. Results show prediction accuracy of about 90% and average PSNR close to that of an oracle predictor with a limited disruption of other data streams' performance.

Chapter 7

Building an Integrated UAV-Network Simulator for UAV-based Mission-critical Applications

7.1 Introduction

Unmanned Aerial Vehicles (UAV) are attracting a considerable attention from the research community. Traditional applications range from surveillance to precision agriculture, environmental monitoring and disaster management [179, 77]. Recent use cases emphasize the role of communications and networking in the overall picture, either to enable intra-swarm coordination or to interconnect the UAVs to ground resources [156, 205]. A new line of contributions deeply integrates UAVs into the communication infrastructure to extend wireless coverage [174].

Clearly, deploying real-world UAV systems poses several challenges, especially in urban or densely populated environments. Thus, simulation is an important option to reduce the

overhead of testing new solutions and architectures. However, there has been a limited amount of effort in developing simulation tools capable to accurately model the fine-grain operations and characteristics of both UAV and networking/Internet of Things (IoT) systems. Mature and popular tools focus on either one of these aspects, often oversimplifying the other. Recent contributions such as AVENS [146] and CUSCUS [214] make a first effort in this direction, interlacing UAV and networking simulation tools. This research builds on these first steps to create a comprehensive simulator – FlyNetSim – capturing the intricate interdependencies between the communication and network environment and UAV operations, such as sensing and navigation, and inner state dynamics (*e.g.*, battery state). We pose a particular emphasis on urban IoT systems, where the UAVs are immersed in a multi-scale technological ecosystem conglomerating several wireless access technologies and communication strategies – *e.g.*, Wi-Fi, Long Term Evolution (LTE) and Device-to-Device (D2D) – as well as backhaul wired networks and computation resources such as edge and cloud servers [52]. Our main objectives are:

• Accurately model UAV operations and dynamics using a software-in-the-loop approach, where the data structures and control pipeline of UAV software are fully preserved.

• Accurately model a multi-scale multi-technology IoT communication environment and its interactions with the UAVs.

• Establish a one-to-one correspondence between UAVs and wireless nodes in NS-3, where the UAVs implement a full network stack supporting multiple network interfaces.

• Preserve individual data paths from and to UAV sensors and controllers.

• Provide a graphical user interface to visualize the status of the system and automatically generate UAV/network scenarios.

• Support emulation, where real-world UAVs can perform on-board simulations of a surrounding network environment while flying.

• Support computation and real-time data processing in-the-loop, for instance to implement and test edge computing architectures.

To accomplish these objectives, we take as starting point two open source simulators – Network Simulator (NS-3) and ArduPilot – and build a fully open source simulation environment for academic research. NS-3 [100] is an extremely popular tool, with a wide community contributing to extend its capabilities. ArduPilot [184] is a widely used software and hardware-in-the-loop simulator, capable of modeling a broad range of unmanned vehicles characteristics in terms of navigation, control and mission planning.

FlyNetSim includes a middleware layer to interconnect the two simulators, providing temporal synchronization between network and UAV operations, and a publish and subscribe based framework [78] to create end-to-end data-paths across the simulators. The middleware architecture we propose is lightweight, and enables FlyNetSim to simulate a large number of UAVs and support a wide range of IoT infrastructures and applications. We illustrate the capabilities of FlyNetSim through several case-study scenarios:

• *UAV control over Wi-Fi*: We used FlyNetSim to evaluate the effect of mobility, delay, and congestion on the operations of a UAV remotely controlled by a Ground Control Station (GCS).

• *Multi-network communications*: We created and tested a case-study scenario in which a UAV uses multiple network technologies for uplink and downlink communications.

• *D2D Communications for UAV swarms*: We evaluate through FlyNetSim the ability of D2D communications to extend network coverage through intra-swarm communications.

• *IoT and Data Streaming*: FlyNetSim supports real-time data streaming from the UAVs. We showcase this ability by testing real-time telemetry and video streaming.

Finally, we also showcase the ability of FlyNetSim to support the "emulation mode", where the simulator runs on a real-world UAV.

7.2 State of the Art

In this section, we provide an overview of UAV and network simulators, and discuss the recently proposed AVENS and CUSCUS.

UAV Simulators - The primary scope of UAV simulators is to model aerodynamics and functional aspects of their control systems. Early simulators, *e.g.*, [8], only simulated human-controlled flight, and autonomous flight and autopilot features were implemented in later tools. Among open source autopilot simulators, Ardupilot and PX4 are most widely used, with the former having a wider range of supported platforms and hardware, including Pixhawk [149], NAVIO [9], and Erle-Brain [5]. Ardupilot is also used by some commercial vehicles, such as 3DR Solo [2].

Several UAV simulators build on top of Ardupilot to extend its capabilities. For instance, Gazebo [121] extends the range of simulated devices to include land rovers, planes, and small robots using the Robot Operating System (ROS) [166]. However, Gazebo is a very heavy-weight software, which may be difficult to run on a single machine when simulating a large number of vehicles. Moreover, ROS is not synchronized with real-time operations and communications of connected applications. XPlane-10, RealFlight and few other commercial simulators are also based on Ardupilot, but mostly focus on motion and navigation. To develop FlyNetSim, we choose Software In The Loop (SITL) [24, 4], one of the simplest simulators based on Ardupilot supporting autopilot navigation.

Network Simulators - Many simulators are available which offer simulation of the full network stack or a portion of it. Most simulators are discrete event-driven, meaning that they form a timeline of events that is sequentially processed, without a strict correspondence between real and simulated time. One of the earliest discrete event-driven simulator is TOSSIM [127] which was used to model communications between wireless sensors. Given our objectives, we are primarily interested in network simulators that can support WiFi and 4G/5G cellular communications. Also, the simulator should support a wide range of protocols and options at different layers of the network.

OPNET [61] is a commercial simulator providing detailed simulation of wireless networks and supports a wide spectrum of protocols and technologies. Among open source network simulators OMNET++ [195] and NS-3 [169] are the most used by the research community. Both simulators are discrete-event driven. Thanks to a wide support from the community, NS-3 can simulate many technologies and embeds a variety of statistical models for channel gain, mobility and traffic generation, and provides a detailed modeling of physical layer operations. Additionally, NS-3 can efficiently interface with external systems, applications and libraries. For the reasons above, we choose NS-3 as the network component of FlyNetSim.

Joint UAV/Network Simulators - Recent work [146] integrates OMNET++ and X-Plane to build a joint UAV-Network simulator – the Flying Ad Hoc Network Simulator (AVENS). The main limitation of AVENS is that the UAV and network simulator communicate via a XML file, which only updates the number and position of the UAVs. This communication model does not support data-paths from and to the UAVs to transfer control, telemetry and sensor data. Also, the UAV and network simulators are not synchronized, thus impairing the ability of AVENS to model and evaluate real-time services and operations. Finally, X-Plane is not an open-source software, thus making the use of AVENS challenging in academic research.

The joint UAV-network simulator CUSCUS [214] has been developed based FL-AIR [7] in combination with NS-3 using tap bridges [18] and containers. CUSCUS uses the communications of NS-3 with Linux containers with real-time scheduling, thus successfully creating network devices corresponding to the UAVs in FL-AIR. However, the current NS-3 implementation has a limited support to interfaces through tap bridges. For instance, this approach prevents the inclusion of important technologies such as LTE and D2D. Also, tunneling through the tap interfaces does not facilitate multi-hop routing through intermediate nodes in the network simulator. These aspects impair the ability of CUSCUS to simulate important multi-technology environments and systems such as the urban IoT. Additionally, in the CUSCUS architecture the UAV control loop with the Ground Control Station (GCS) is discorporated from the network simulation. So, the simulator cannot be used to evaluate the effect of the network environment dynamics on the operations of the UAVs. Finally, the use of containers to run the UAVs imposes a larger computational complexity compared to other options. This aspect is particularly relevant as if NS-3 event processing is delayed compared to the UAV simulator clock due to a large number of network events, then synchronization will break down. Therefore, an approach based on containers severely limits the complexity of the simulated environment.

7.3 General Framework

As indicated in the previous section, we choose NS-3 and ArduPilot as the two main blocks of FlyNetSym. However, interconnecting the two simulators is non-trivial. One of the key desiderata is to build a lightweight simulator capable to include a large number of UAVs and other devices, and a complex communication infrastructure. In the following, we discuss some of the available options and the chosen approach.

Virtual Machines - A possible approach is to run each UAV in an individual Virtual

Machine (VM) and connect the VMs via the network simulator running on the native host. However, this approach results in a considerable resource usage, which impairs scalability. Moreover, communication through the upper layer of the protocol stack of the host to the VMs introduces some delay and necessitates additional memory usage. Finally, due to the isolation of the environment embedding each UAV, their synchronization would require the implementation of a complex algorithm.

Containers - The second option is to use lighter-weight containers instead of VMs, thus mitigating the scalability issue. Note that containers are used in CUSCUS. However, containers share the synchronization issue with the VMs. Moreover, the containers need to be connected to NS-3 via tap bridges (see the approach in CUSCUS [214]) associated with a "ghost node" in NS-3. However, in the current implementation of NS-3, this communication model does not support several key technologies, thus limiting the range of possible simulations. Additionally, the use of containers would prevent "on-board" simulations, as the autopilot hardware would need to run within a container.

Multi-threading - The option we select for the development of FlyNetSim is multi-threading, where each UAV simulator module corresponds to a thread. This model provides improved scalability and seamless synchronization between the UAVs and the network simulator. Additionally, on-board simulation will be possible, as the network simulator and the UAV run on the same host.

We remark that in this model we can directly create a unique node in NS-3 corresponding to each UAV and the GCS, without the need to have tap interfaces. Also, each UAV (NS-3 node) can be equipped with multiple network interfaces for multi-technology communication.

The main challenge we face is the establishment of an effective communication between each UAV simulation module. A communication model based on files or shared memory makes the construction of individual and end-to-end data paths difficult due to the need for

sophisticated mapping. This especially applies to a case where heavy-weight data streams need to be transported – such as video or data streams – a case in which file or memory sharing approach would grant inferior efficiency and performance. We overcome this issue by incorporating Zero Message Queue (ZMQ) in the framework – a fully open source based on a publish and subscribe communication model.

ZMQ-based communication model: ZMQ is a light weight, extremely fast messaging library written in C/C++ with binding support in various languages. This flexibility makes it a widely accepted tool to support cross-platform IoT applications. The sockets in ZMQ are created by the publishers and subscribers of the messages in the message queue of a broker. Thus, the publisher to subscriber connections are non-blocking and fully asynchronous. ZMQ also supports message filtering at the subscriber.

Importantly, ZMQ supports many-to-many communication between end-points. This allows the establishment of sensor- and control-specific streams of traffic between the UAVs. However, it may result into an increased delay due to interleaving of packets in the queue, which in turn would result in an increased total end-to-end delay of the ZMQ communication. In order to assess this aspect, we perform an evaluation of the end-to-end delay of ZMQ defined as:

$$D_{ze2e} = D_{pub} + D_q + D_{sub} \tag{7.1}$$

where, D_{ze2e} is the delay between a ZMQ sender and receiver, D_{pub} is the publishing (sender to the ZMQ), D_{sub} is the reception delay (ZMQ to the receiver) and D_q is the queueing delay in the ZMQ.

We measure the total delay in two scenarios: *(Single P/S)* the streams are sent over a single ZMQ publisher and subscriber pair or *(Parallel P/S)* each data stream is matched with a ZMQ publisher and subscriber pair. Fig. 7.1 shows that the end-to-end delay as a function

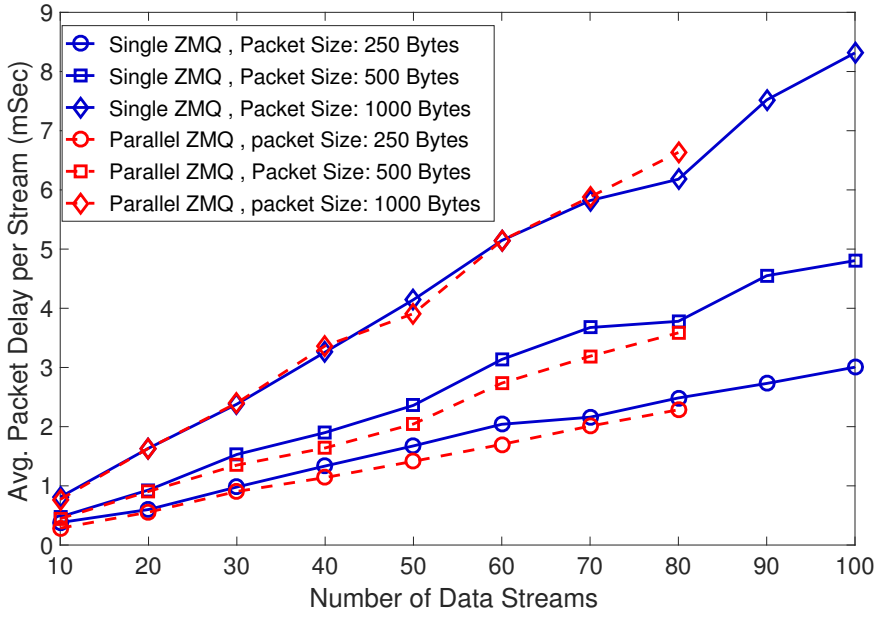


Figure 7.1: Average total ZMQ delay as a function of the number of nodes using a single or multiple ZMQ publisher/subscriber pairs.

of number of data streams N. As expected, the total delay increases with N. The single and parallel strategy have comparable delay, and in both cases the end-to-end packet delay is less than 10 milliseconds even for large values of N (100 streams) and data packet size (1000 bytes). Thus, both options have sufficiently small delay to support the simulation environment. However, a disadvantage of the parallel strategy is that each data stream has an individual socket pair, and a corresponding file descriptors for each of ZMQ end. In our preliminary study, we could create up to 80 parallel ZMQ publisher/subscriber pairs. Note that there is default limit of a single machine that restricts the number of file descriptors. The default value can be increased at the cost of an increased memory usage. Herein, we use a single ZMQ publisher/subscriber pair per communication direction for multiple data streams. This choice can support hundreds of concurrent data streams with lower complexity in implementation.

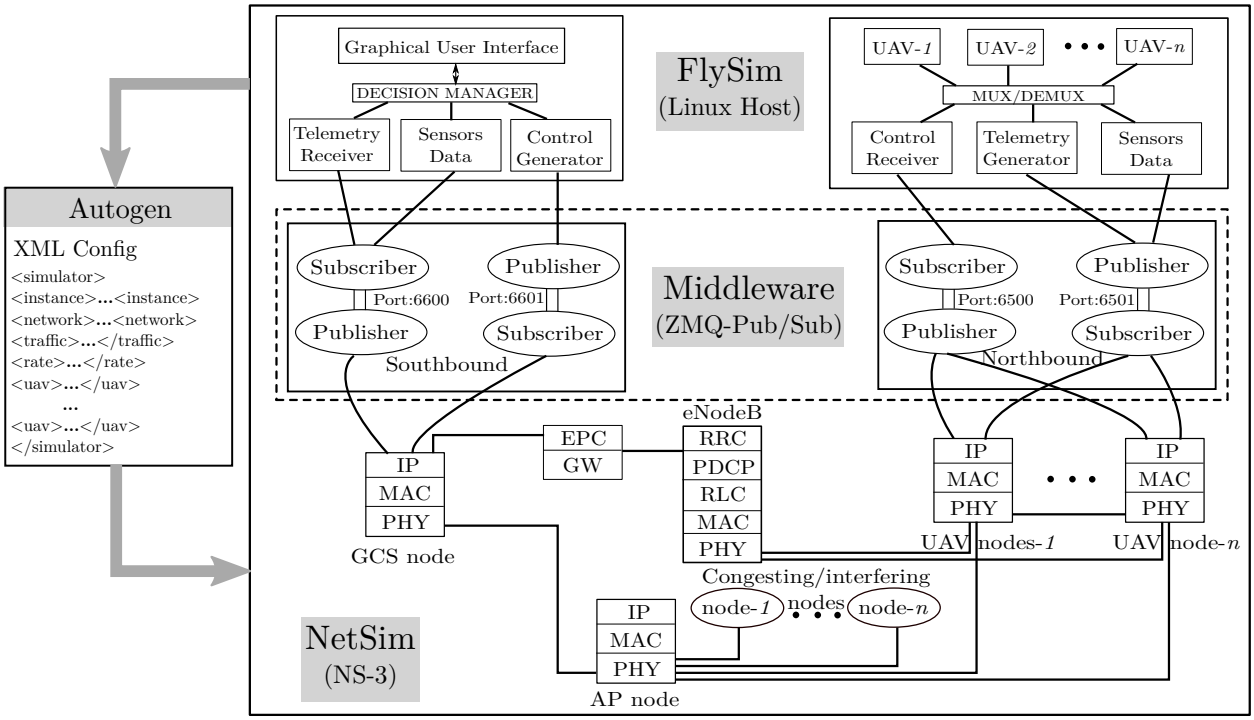


Figure 7.2: Architecture of FlyNetSim.

7.4 Architecture

The architecture of FlyNetSim consists of the following major components: the *FlySim* component that initiates the UAVs, GCS, and the visualization tools; the *NetSim* component simulates the end-to-end network infrastructure and environment (including non-UAV nodes); the *Autogen* component creates a map between the UAV components (including sensors) and the corresponding network nodes, and the interfaces and other infrastructure needed by the NetSim; and a *Middleware* component supporting ZMQ-based bidirectional communications through the network. In the following, we describe in detail these components.

1) FlySim: FlySim is the top layer of the architecture, and contains the UAV and GCS node(s) running on the native host machine. The module instantiates the number of UAVs defined in the user input. Each instantiated UAV corresponds to a simulated vehicle in the

SITL simulator based on Ardupilot. Thus, each vehicle is accompanied by a vector state indicating variables such as position, battery state, telemetry and sensor information. The GCS side contains a custom graphical user interface (GUI) to configure and control the UAVs to be created, and track their positions and other information from the telemetry.

2) NetSim: The NetSim component creates an interface with NS-3, integrated with ZMQ and other necessary libraries based on the configuration of communications in the simulated environment. The interface creates nodes such that there is one-to-one correspondence between each UAV in Flysim and a node in NetSim. Moreover, it enables the UAVs to be equipped with multiple wireless interfaces, *e.g.*, WiFi and LTE, D2D. NetSim also configures the network environment in terms of congestion and interference from nodes outside of FlySim based on user input. The user can define the number of interfering nodes, their position, traffic rate, data statistics and technology. The module allows the use of any wireless protocol or strategy supported by NS-3, including ad-hoc point-to-point communications between the UAVs or infrastructure based communications over technologies such as LTE or WiFi. Finally, the NetSim module also handles mobility within the wireless network based on the actual mobility of the UAVs retrieved from the FlySim.

3) Autogen: The Autogen component is responsible for automatic code generation in the FlySim and NetSim based on the input defined by the user in the graphic interface at the GCS. The module creates an XML file based on the input, which is passed to the NetSim as its arguments to create the nodes.

4) Middleware: The middleware interconnects the FlySim and NetSim via communication over ZMQ, creating the end-to-end data path for individual data streams from the GCS to the UAVs and vice versa. As both the GCS and the UAV application softwares are located outside the NetSim, the middleware needs to be used at both ends of the NetSim. We name the parts of the middleware as north-bound and south-bound middleware. The north-bound

part connects the NetSim to the UAVs, whereas the south-bound connects the GCS to the NetSim. As described in the previous section, we use one ZMQ publisher/subscriber pair for communication in each direction for each of north-bound and south-bound middleware. The middleware is also responsible for synchronization between the FlySim and NetSim block as described in the following.

7.4.1 Synchronization

NS-3 is a discrete event-driven simulator, whose scheduler typically executes the events sequentially without synchronizing with an external clock. On the other hand, ArduPilot is a real-time operator, where the temporal evolution of the UAVs' state follows the external clock. To solve this disparity, we use the real-time scheduler in NS-3, which aligns the processing of scheduled events with their actual time. However, we remark that in dense simulations the scheduler may actually be "late" with respect to the clock. In this case, either synchronization is lost ("Besteffort") or late events are discarded ("Hardlimit").

We embed in FlyNetSim middleware's additional steps to ensure synchronization between the two simulators by timestamping each packets flowing through NS-3. Fig. 7.3 shows the message flow across the different layers of the simulators to preserve synchronization between the network and UAV operations. Let t_0 be state of the system clock at the time when a packet enters the NS-3 environment, and Δ the difference between the submission and reception time, that is, the network propagation time. When the packet exits NS-3, we acquire the system time t_1. If $(t_1 - t_0)$ is smaller than Δ, *i.e.*, the network event processing is faster than real-time, the middleware waits an additional $(\Delta + t_0 - t_1)$ interval to release the packet to ArduPilot.

In the case in which the NS-3 scheduler falls behind real-time, we propose to "*freeze*" the UAV simulator to match the elapsed time with the current clock time. Note that this is critical to

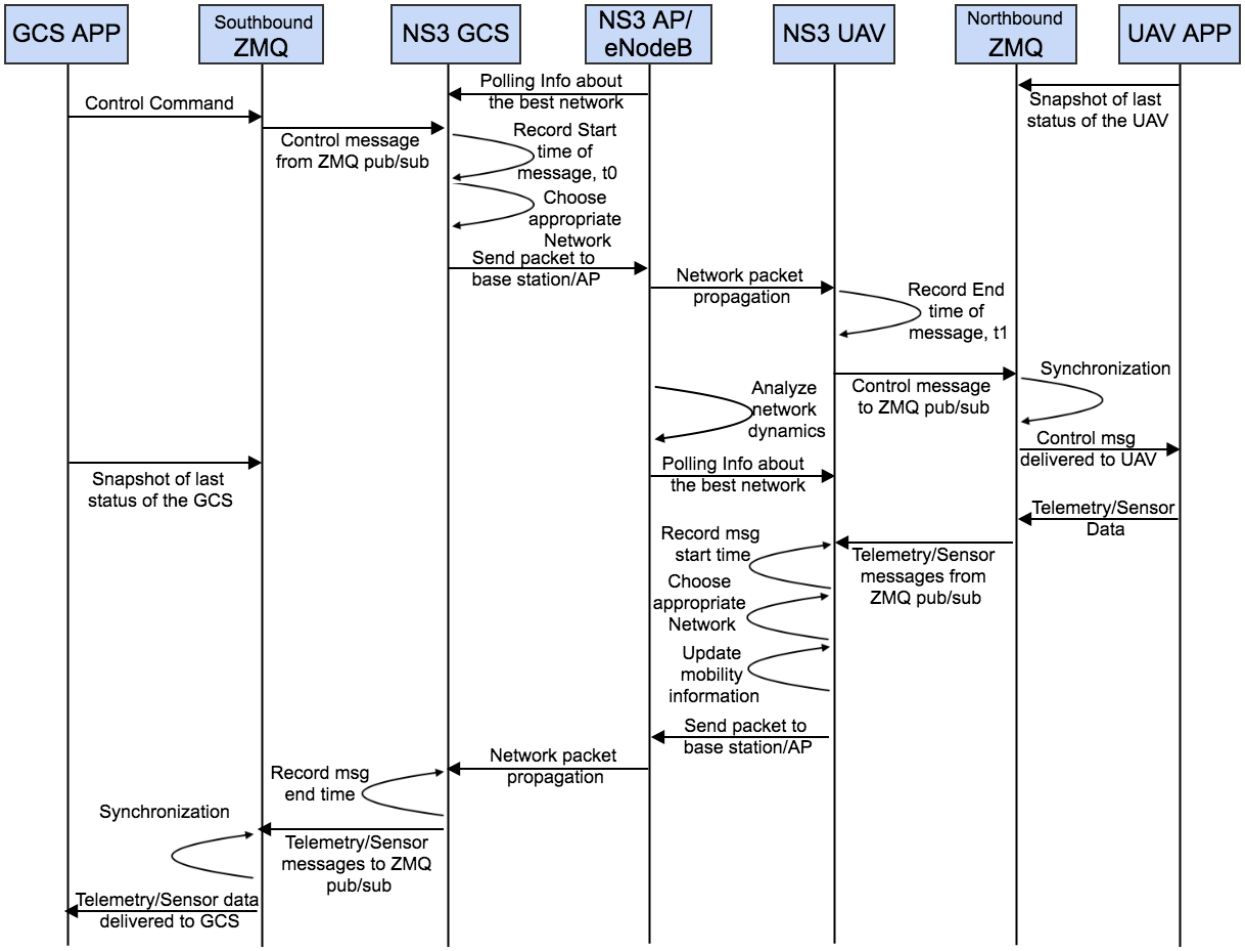


Figure 7.3: Message flow across the FlyNetSim architecture

preserve an accurate reconstruction of flight dynamics and inner state (*e.g.*, residual battery charge) evolution with respect to network events. The idea is to take a snapshot of the vehicle state after the completion of last command and recreate the simulation at the correct time. This will avoid the simulator performing actions for a longer time than prescribed.

7.4.2 Mobility

FlyNetSim matches the actual dynamics of the UAV in the FlySim with the position of the corresponding wireless nodes in NS-3. This is realized by intercepting the variables indicating the position, speed and direction in the telemetry flow at its creation, thus avoiding the,

possibly large, time needed to receive the actual telemetry packets through the network. Note that telemetry is not created in response to commands, but, instead, in response to any update in the state of the vehicles. This ensures an accurate matching between the two simulators, where the network counterpart has the same granularity of representation of UAV dynamics as the UAV simulator. Note that non-UAV nodes in NS-3 can use available mobility models, such as constant position, constant speed, random walk, *etc.*

7.4.3 Implementation

The implementation of FlyNetSim is based on C/C++ and Python. Specifically, we integrate the "libzmq" Python library with the UAV simulator and the "libczmq" library bindings with NS-3. The network simulator is realized with two parallel threads which continuously listen for messages from the GCS and UAVs, respectively. The module within the NS-3 code that sets the position of the UAVs based on the telemetry data received from the external ZMQ in the listening thread, is based on the Haversine formula [197].

On the application side, each UAV is instantiated using a thread running the UAV simulator based on ArduPilot. We use Dronekit [1] to operate the UAVs following the Micro Air Vehicle Link (MAVLink) protocol [148] that includes pitch, roll, and yaw movements. The custom GCS is created along with a GUI written in Python that can be easily integrated with "pymavlink" libraries as well as receiving telemetry informations using MAVLink.

7.5 Case Studies and Numerical Evaluation

We provide a series of case-study scenarios to illustrate the capabilities of FlyNetSim to capture fine-grain interdependencies between communication/networking and UAV operations, as well as to simulate a wide range of rich network and IoT environments.

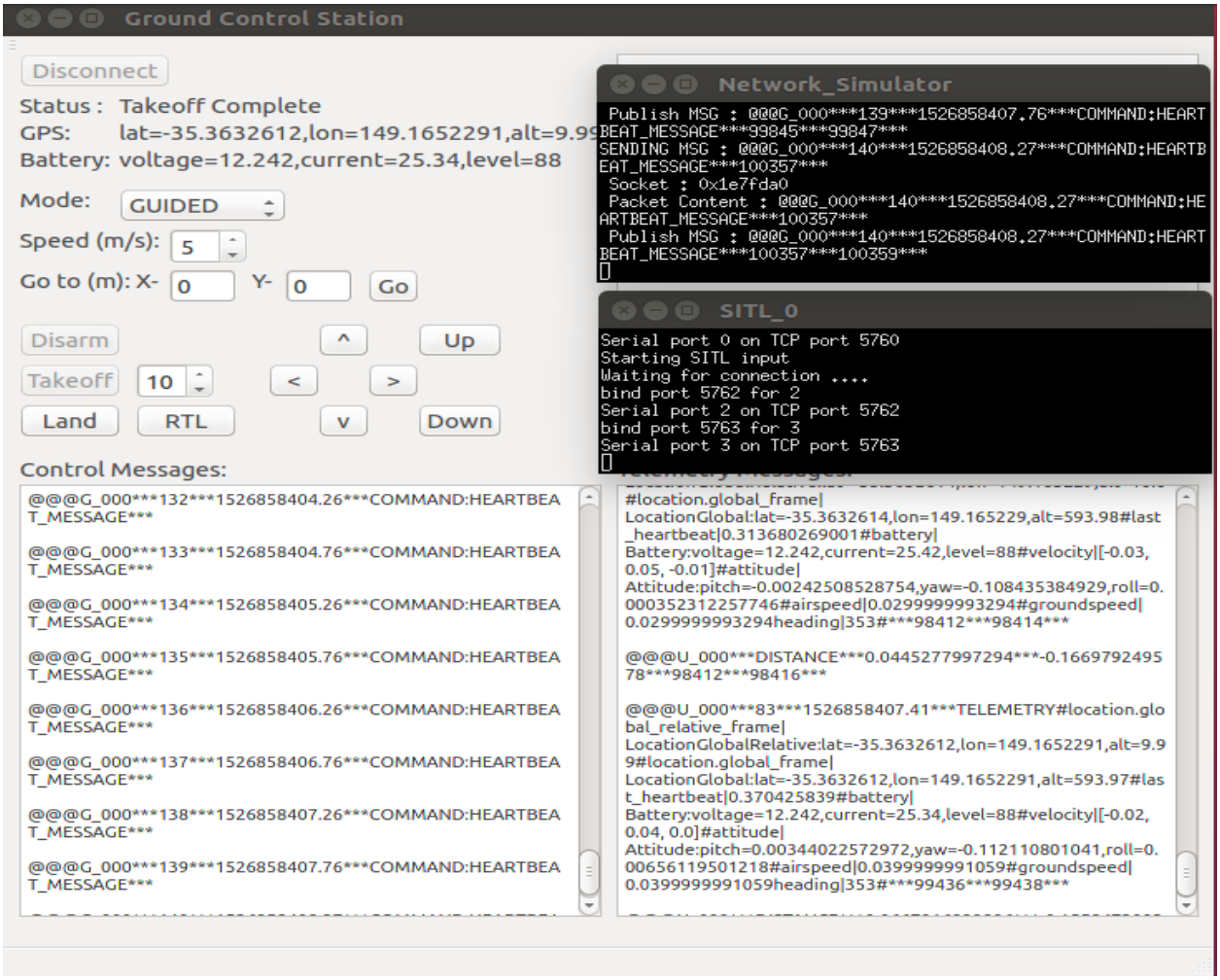


Figure 7.4: GUI Panel for Controls and Telemetry logs

7.5.1 Case Study I: Single UAV over WiFi

In this first case study scenario, we consider a single UAV communicating with the GCS over WiFi. This simple scenario has been modeled using other integrated UAV/network simulators to evaluate the effect of mobility on bidirectional communications. However, FlyNetSim allows the inclusion of other nodes injecting traffic into the network, thus creating congestion.

First, we assume the UAV hovers to maintain a constant position at a fixed distance from the WiFi Access Point (AP). The simulation accurately models the control messages sent

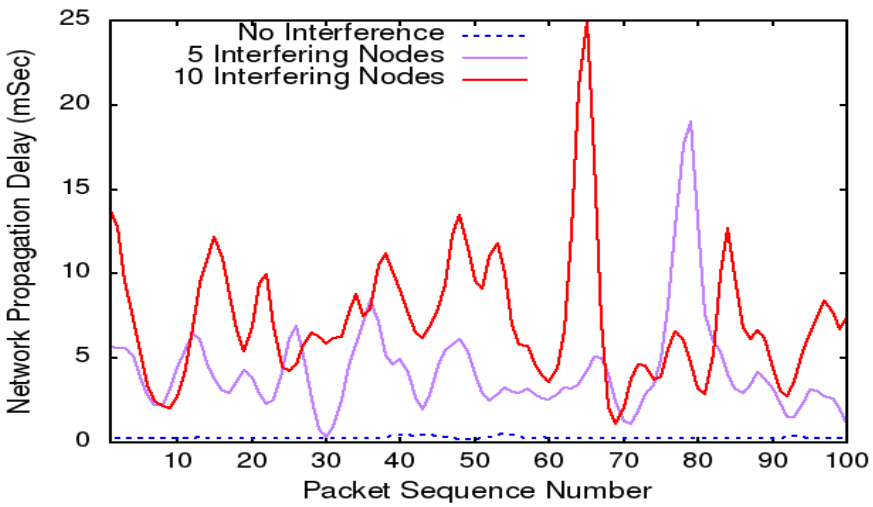


Figure 7.5: Temporal trace of network delay in response to variations in channel quality and availability due to contention.

from the GCS to a single instance of UAV through the GUI panel as depicted in Fig. 7.4. The figure also shows the control and telemetry logs on the GUI. In the absence of other traffic, the average packet propagation delay through the network is approximately 0.24 ms.

We measure the delay in the presence of nodes contending for the channel resource in the WiFi network, where each node has a traffic arrival rate of 10 Mbps and each packet is 1000 bytes long. Fig. 7.5 shows a temporal trace of the network delay of control packets generated at the GCS and propagating through the network to the UAV. We use adaptive data rate for the WiFi channel in NS-3. In the absence of interference, the total delay is below 1 ms. When the number of contending nodes is set to 5 and 10 the delay increases to an average of approximately 4.20 ms and 7.06 ms, respectively. Note that the delay presents significant variations over time due to channel contention and adaptive data rate, where the delay variance increases with the number of contending nodes. This effect may create undesirable jitter in the reception control packets. Fig. 7.6 depicts the average delay of control packets reception as a function of the number of nodes contending for channel access.

In the same setup, we test the feature of FlyNetSim matching the position of the UAV in

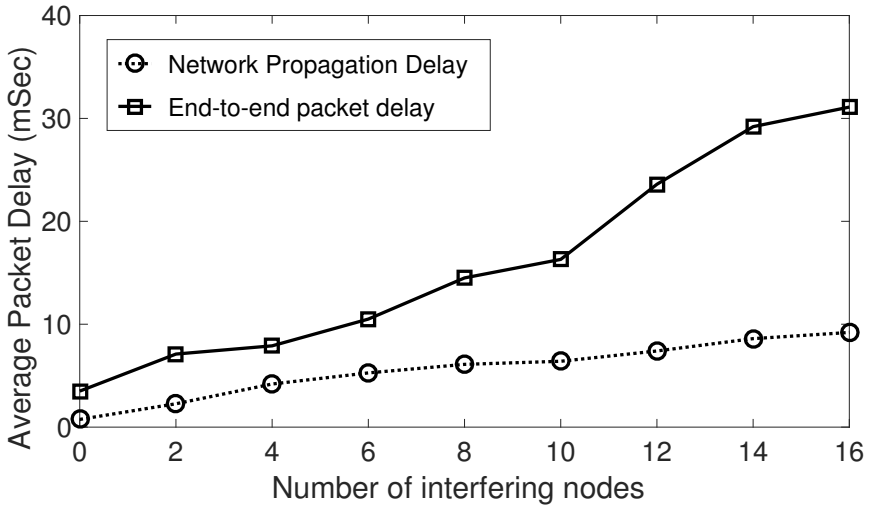


Figure 7.6: Network and end-to-end delay as a function of the number of nodes contending for the channel (excluding the UAV).

NS-3 to that in the UAV simulator. For this experiment, we use random movements of the UAV from the GUI panel, and receive updated latitude and longitude values from the telemetry. The NetSim component of FlyNetSim converts the position in distance from the other nodes. We measure the received signal strength (RSS) of the UAV node from the AP. The blue line in the Fig. 7.7 shows the trajectory of the UAV over time in the XY plane and the dotted red line shows the corresponding decrease in the RSS value. The fine-granularity correspondence of the RSS in the simulator to the position of the UAV determined by its operations enables the development and testing of physical and link layer protocols.

7.5.2 Case Study II: Multi-Network Environment

FlyNetSim seamlessly supports heterogeneous wireless network environments, as well as multiple wireless interfaces at each UAV. To the best of our knowledge, FlyNetSim is the only integrated simulator including this feature.

To illustrate this capability, we implemented a multi-network environment including WiFi and LTE networks, where each UAV has a WiFi and an LTE interface and the corresponding

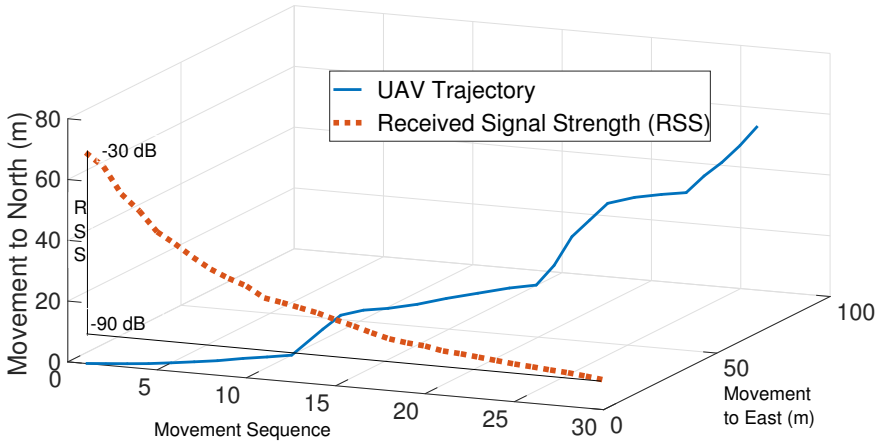


Figure 7.7: Variation of received signal strength (WiFi) in response to the motion of the UAV.

protocol stack. In this use case, the GCS can be placed at the network edge, *i.e.*, connected to the WiFi AP and the LTE eNodeB. Herein, we characterize the GCS as a remote host connected to the Evolved Packet Core (EPC) gateway of the LTE network to support long-range communication. The autogen module of the simulator creates the multiple wireless interfaces as defined by the input.

We focus on a scenario where the WiFi network is used to transport control messages from the GCS to the UAVs, and the LTE network is used to transport telemetry and high volume sensor data traffic from the UAV to the GCS. Note that case-studies where the network is dynamically selected based on Quality of Service (QoS) requirements can be implemented.

In the shown experiments, WiFi nodes use adaptive data rate with maximum rate of 54 Mbps. The LTE network is created using different uplink and downlink frequency bands of 20 MHz each, and the LTE Frequency Division Duplex (LTE-FDD) mode is used for transmission. Nakagami channel model is used to model LTE signal propagation [209]. Specifically, we adopt the trace-based Extended Pedestrian A (EPA) fading model as indicated in the 3GPP standard [136]. The LTE network determines the Channel Quality Index (CQI) and assigns the corresponding Modulation and Coding Scheme (MCS) for transmission. Fig. 7.8 depicts

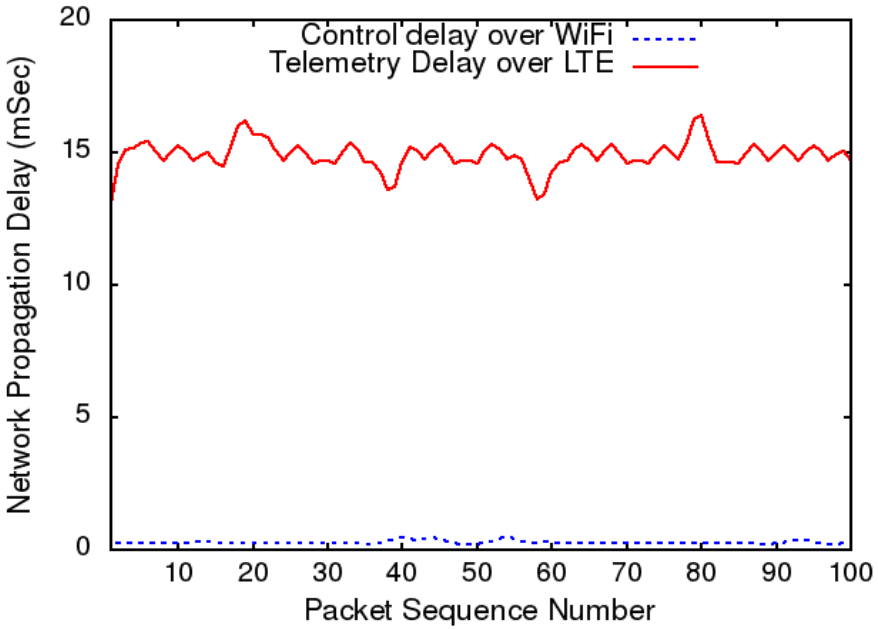


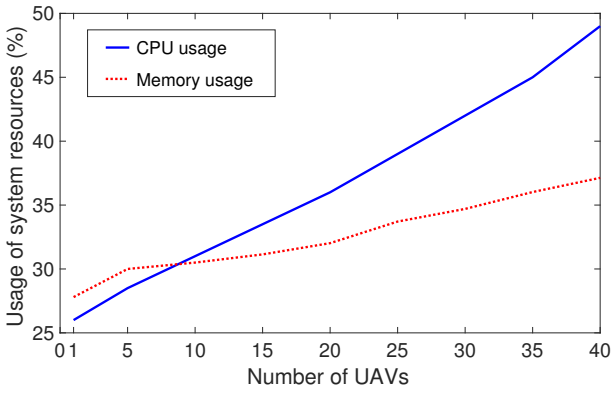
Figure 7.8: Network delay of control messages (WiFi) and telemetry (LTE).

the end-to-end delay of the control data (transmitted over WiFi) and telemetry data (transmitted over LTE), which average at approximately at 0.24 ms (control packets) and 15 ms (telemetry). We remark that telemetry information is extracted before its transmission and immediately used to update the position of the UAV, which determines the channel gain of both LTE and WiFi channels.

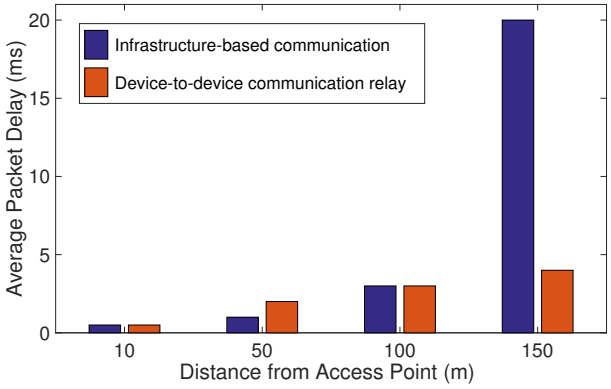
7.5.3 Case Study III: Multi-UAV

This case-study scenario focuses on the simulation of UAV swarms. First, we compare the scalability of the proposed approach with that of the approach adopted by AVENS and CUSCUS. Note that the number of UAVs in AVENS is limited to 20 due to restrictions in X-Plane, whereas CUSCUS does not have any strict limit, and was used to simulate up to 30 UAVs.

We tested the scalability of FlyNetSim by instantiating multiple UAVs with basic functionalities, including the transmission of control messages from the GCS to each UAV and



(a) Percentage of system resource usage as a function of the number of UAVs instantiated in the simulator.



(b) Network delay as a function of the distance between the closest UAV in the line topology and the WiFi access point.

Figure 7.9

telemetry on the reverse path. We run the simulator on a laptop with 16 GB of RAM and an Intel Core-i7-6700HQ processor. We could instantiate up to 70 UAVs communicating using a WiFi network. The maximum number of UAVs is 40 when LTE network is used, due to limitations imposed in NS-3 in connection with the 40 ms periodicity of transmissions. However, the value can be increased up to 380 based on the standard. Our architecture does not impose any restriction on the number of UAVs, and is only constrained by the available system resources. Therefore, more than 70 UAVs can be instantiated with a more powerful computer.

We measure the system resource consumption in terms of memory and CPU usage as a function of number of UAVs instantiated. Fig.7.9a shows that both the CPU usage and memory usage increase linearly with the number of UAVs. A direct comparison with AVENS simulator is not possible. However, we compare FlyNetSim resource usage to that of CUSCUS, which is built on network containers: FlyNetSim almost halves CPU usage and substantially reduces memory usage.

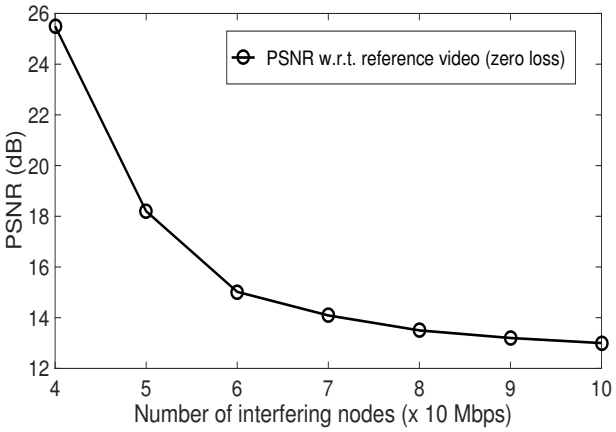
We further demonstrate the capabilities of FlyNetSim by instantiating a use-case where the UAV swarm collaboratively transports packets to extend network coverage. The UAVs use

D2D communications to form an ad-hoc network and relay messages from and to the GCS. In the experiment we created 4 UAVs positioned in a line topology where the UAVs are 50 m apart from each other. We then measure the network delay as a function of the distance between the WiFi AP and the closest UAV. Fig. 7.9b shows that if packets are directly transmitted to the UAVs the average packet delay has a considerable delay as the UAVs approach the border of network coverage. Intra-swarm D2D communications considerably extend coverage. Note that a similar scenario was shown in CUSCUS, where the UAVs broadcast the messages to the swarm, instead of forming an ad hoc network based on D2D communications.

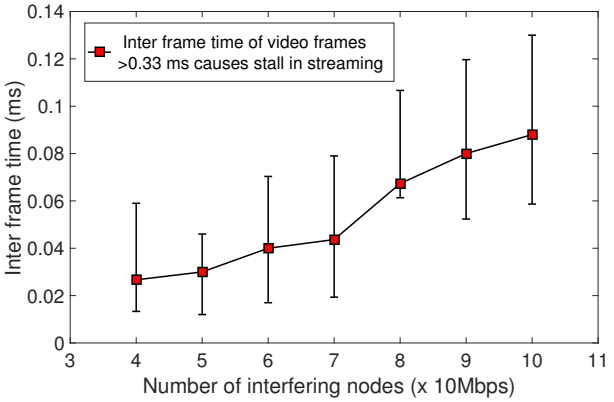
7.5.4 Case Study IV: IoT Applications

Unlike existing integrated network/UAV simulators, FlyNetSim not only aims at network measurements and UAV control, but also provides a comprehensive framework for the evaluation of IoT applications on UAVs. The end-to-end data-path enables the network simulator to receive and analyze sensor data from the UAVs and measure the QoS granted by different network scenarios and technologies. We observe that FlyNetSim can also integrate real-time processing of data, thus enabling the exploration of edge-assisted IoT architectures.

We implement and test a scenario where FlyNetSim is used to support streaming of real-world encoded videos over the network simulator, where packet loss creates artifacts and impair the performance of processing algorithms. To the best of our knowledge, all available integrated UAV simulators do not support end-to-end encoded data transmission, especially over an articulated network infrastructure and a wide range of scenarios. In the experiments we used a H.264 AVC [203] encoded video of 624 frames with a resolution of 480 pixels and frame rate of 30 frames per second. We used the MPEG Transport Stream (TS) container for the transmission of the video. We set the network so that in the absence of contending



(a) PSNR as a function of the number of nodes contending the channel in the WiFi network.



(b) Inter frame time of video frames at the decoder as a function of varying number of nodes contending the channel in the WiFi network.

Figure 7.10

nodes packet loss has very low probability and the quality is delivered with perfect quality. Fig. 7.10a illustrates the degradation of the Peak Signal-to-Noise Ratio (PSNR) at additional nodes contending for the channel resource are introduced in the network. With 10 additional nodes, the video stream incurs a PSNR degradation of more than 13 dBs. Fig. 7.10b shows the inter-frame time defined as the time lapse between consecutive frames. The video in the experiment has a frame rate 30 per second, thus, the expected average inter-frame time is 0.033 ms. As the number of nodes contending the channel resource increases, the inter-frame time increases compared to the expected value.

7.6 Emulation Mode

FlyNetSim supports an "emulation mode", where a real UAV can communicate with simulated resources through the network simulator. This mode still uses all the components of the simulator, including the GCS and GUI panel, the middleware and NetSim, but the simulated UAV vehicle is replaced by real-world UAV. We demonstrate the emulation mode by connecting a UAV 3DR Solo via USB serial to the laptop running the FlyNetSim as

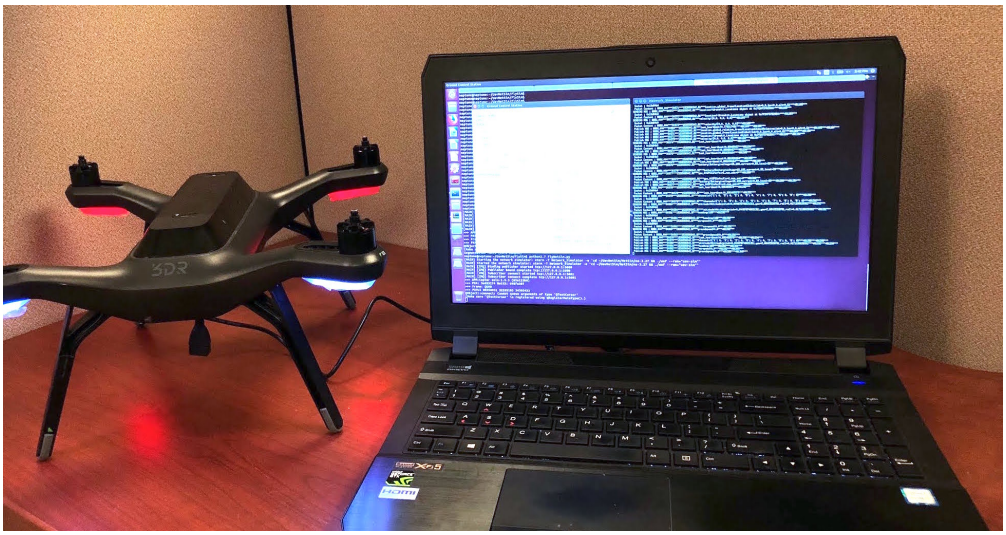


Figure 7.11: Emulation mode where FlyNetSim is run on-board a 3DR Solo UAV.

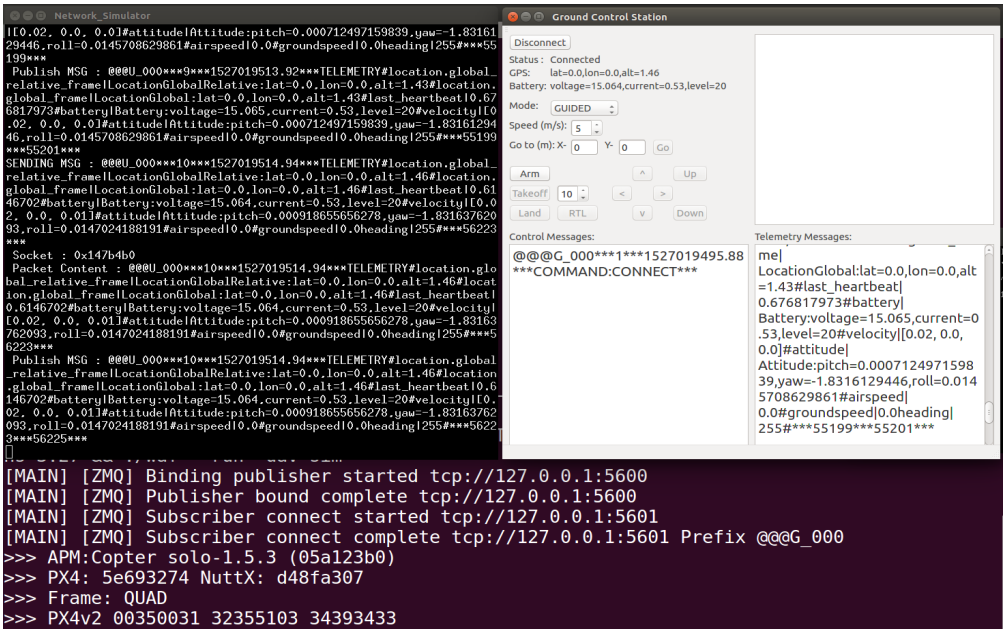


Figure 7.12: Emulation telemetry output from 3DR Solo UAV

shown in Fig. 7.11. A similar configuration would enable on-board simulations during flight, e.g., using a compact platform like Raspberry Pi instead of the laptop. Fig. 7.12 shows the GUI panel as telemetry regarding battery status, GPS location, etc., is received through the simulated network. The network simulator panel shows the log of the packet flow across the simulator from the UAV to the GCS. Note that the experiment was conducted indoor, where the GPS signal is not available, and thus the latitude, longitude values are shown as zero.

7.7 Conclusions

The primary contribution of this work is a fully open source, integrated UAV-network simulator capable to support a wide range of network scenarios and use-cases while accurately modeling UAV operations. The architecture uses a lightweight custom built middleware to effectively simulate the UAV network with closed loop, synchronized, end-to-end data-path, considerably reducing system resources usage compared to other available integrated simulators. Additionally, the support for emulation mode enables experimental research using real-world UAVs and sensor devices over simulated complex environments such as the urban IoT. We demonstrated the capabilities of the proposed simulator through a series of case study scenarios, including multi-technology networking, intra-swarm D2D communications and IoT data streaming.

Chapter 8

Infrastructure Assistance to Autonomous UAV Systems

8.1 Introduction

Unmanned Aerial Vehicles (UAV) have been the center of significant recent attention from the research community. The traditional focus on robotics aspects of these interesting airborne systems, such as flight dynamics, autonomy, navigation and group coordination [101, 115, 119], is being complemented by a growing body of work on innovative communication, networking and signal processing techniques supporting their operations [95, 20, 112].

Along the latter line of inquiry, a recent trend interconnects the UAVs to the Internet of Things (IoT) infrastructure to augment their capabilities. For instance, cellular networks can increase the communication range of the UAVs, and grant them access to the infrastructure. Edge or cloud servers can take over the execution of heavy-weight computing tasks, and coordinate the operations of multiple autonomous UAVs operating in the same area [57]. Finally, through the infrastructure, UAVs can access data streams from external sensors,

thus increasing their sensing accuracy and range [174].

The objective of this research is to assess the main challenges and trends in establishing an effective interconnection between the UAVs and the infrastructure. Intuitively, the degree of "trust" the UAV can afford to give to the infrastructure depends on a number of Quality of Service (QoS) metrics heavily influenced by network and environment parameters, such as the used protocol suite and technology, network load, building density, and UAV motion characteristics. We explore a wide range of scenarios and identify parameter regions where remote control or assistance through the IoT infrastructure is feasible. We focus our investigation on two case-study applications: (*i*) The UAVs transmit telemetry to a remote controller at the network edge to inform navigation control; (*ii*) the UAVs offload computing tasks to an edge server.

In the former case, network impairments may impact delay or loss of telemetry packets, influencing the estimation error of UAVs' position or speed. An excessive estimation error may affect the ability of a remote controller to, for instance, determine disjoint navigation trajectories for multiple UAVs operating in the same area.

In the latter case, which refers to edge computing paradigm [52], the UAVs transfer data to be processed to an edge server connected to the communication infrastructure. Clearly, the tolerable delay and delay variations, as well as the frequency and distribution of trains of data packets to be delivered to the edge server depend on the level of support the UAV asks to the infrastructure. For instance, the offloading of sporadic, although heavy-load, tasks at the application layer may not impose extremely stringent requirements to the network. Conversely, the offloading of the processing of signals used for fine-grain navigation control, may require a tight distribution of data delivery timing to result in an effective control.

In order to capture the characteristics and complexity of infrastructure-assisted UAV systems, we use the FlyNetSim simulator we presented in [38, 39].

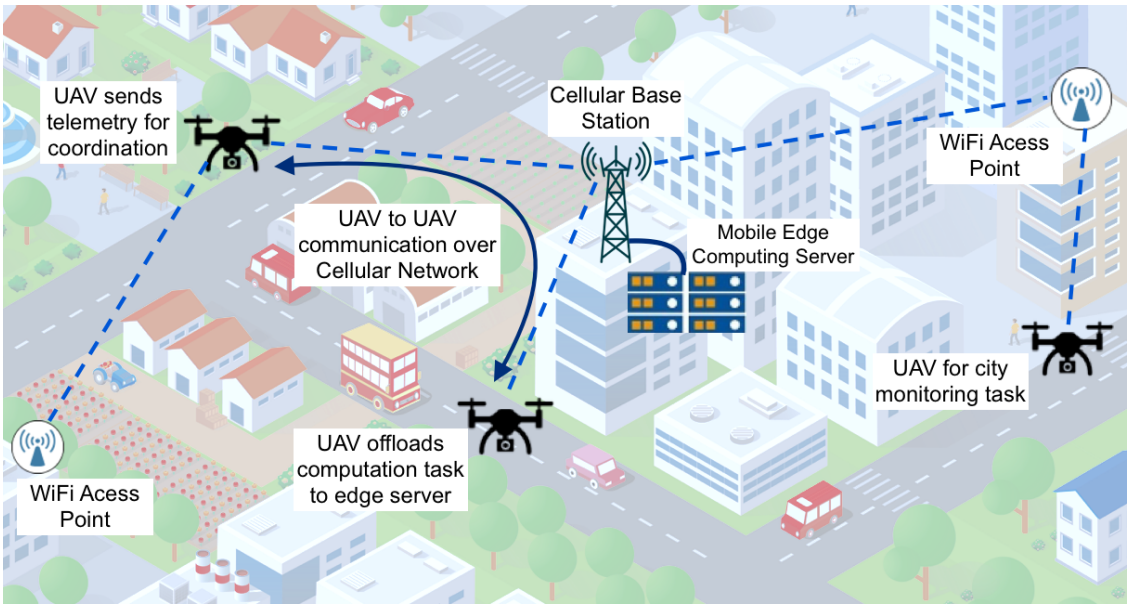


Figure 8.1: Scenario considered: UAVs connect to the communication/processing infrastructure to enhance their capabilities.

Our results emphasize how the performance of infrastructure-assistance are heavily influenced by the characteristics of the environment, such as the number of nodes active in the network, the distance, and the characteristics of the application and its corresponding traffic emission pattern. We conclude that effective infrastructure-assistance can be only achieved by enabling UAVs to use multiple technologies, and by implementing algorithms capable to switch from one to another during a mission.

8.2 Related Work

Autonomous UAVs necessitate to execute algorithms analyzing in real-time information rich signals, and extend their communication range to interconnect with remote flight coordinators or other UAVs. In the literature, many solutions and frameworks are available. We summarize the key approaches by dividing them into three categories: *(a)* Infrastructure-assisted communications; *(b)* Infrastructure-assisted computing; and *(c)* Flying ad-hoc networks.

8.2.1 Infrastructure-Assisted UAV Communications

Clearly, urban area are at the same time the most challenging and infrastructure-rich environments. Thus, the UAVs have the opportunity to obtain help from available base stations and users, but also face many communication issues at different layers, including poor and/or unstable signal gain and contention from other active users.

Several contributions address the problem of interconnecting the UAVs with the infrastructure. [94] proposes to use properly aligned directional antennas to extend the range of UAVs communications over WiFi. An experimental evaluation of UAV communications based on several WiFi standards was presented in [98]. However, the aforementioned works do not consider UAV specific applications and the impact of UAV-specific characteristics on communication performance. In [174], the authors presented a multi-hop communication strategy with dynamic make-before-break mechanism. The framework uses probes to evaluate available paths and select the one providing performance matching the needs of the UAV.

Cellular infrastructure-based UAV communications were proposed in [215], where aerial and ground UEs coexist in the same area. To increase the reliability of the cellular connection of the UAVs, an interference cancellation approach was used in [157]. A communication strategy over unlicenced bands was derived in [28], where a regret function is defined to learn optimal duty cycle selection and support the coexistence of WiFi with other active technologies. Different from these contributions, herein we study the feasibility of using the infrastructure for relevant UAV based applications over available network technologies and different channel conditions.

8.2.2 Infrastructure-Assisted Computing

To mitigate the limitations of on-board computation on the UAVs, the infrastructure can assist by taking over computation tasks necessary for autonomous UAV operations. By placing compute-capable devices at one – wireless – hop distance from mobile devices, edge computing offers low-latency services compared to traditional cloud computing. Remarkably, offloading to edge servers not only has the potential for improving computing quality – for instance by executing more complex algorithms – and/or delay compared to on-board options, but also is a natural platform to support coordination across multiple UAVs.

Several contributions address challenges related to edge computing for UAVs. A framework to determine trajectories optimal both from the point of view of offloading and mission was proposed in [60]. The framework also accounts for constraints on the speed of the UAV. A hierarchical offloading approach was presented in [196] with real-world UAV-embedded setup for a computer vision based computing application. A framework for UAV-cloud computing for disaster rescue applications was proposed in [138]. In this work, we investigate the impact of the network and network parameters on task offloading.

8.2.3 Flying Ad-Hoc Networks

When the UAV operates beyond the range of ground resources, Flying Adhoc Networks (FANET) [45] are a suitable technologies to interconnect the UAVs and allow cooperation and coordination. Importantly, the low latency of direct UAV-to-UAV communications may improve the ability of a swarm to coordinate.

Note that due to UAV mobility, frequent reconfiguration might be needed when ad hoc communications are used for range extension or relaying of data streams. Despite this issue, using intermediate relay nodes can improve the communication quality and range,

especially as UAVs can form 3D topologies to avoid the impairments that characterize urban environments.

One of the main challenges in achieving effective communications is the distributed nature of flying ad-hoc networks. This may impair performance especially in dense deployments. To improve the performance of FANETs, [206] proposes to use millimeter-wave cellular communications with beamforming. However, the implementation of such model on real-world UAV is extremely challenging, especially due to the rapidly changing topology induced by moving UAVs.

8.3 Infrastructure-Assisted UAV System

We consider an UAV operating in an urban environment, where multiple wireless access networks coexist. Specifically, we focus our attention on widely used communication technologies, that is, Wi-Fi and Long-Term Evolution (LTE) networks, which may support the intense transfer of data necessary to assist UAV operations. We do not assume that a channel is dedicated to the traffic generated by the UAV but, instead, include in the environment the activity of wireless nodes operating on the same channel resource. This will enable the study of the interactions between data streams induced by networking protocols at the different layers of the stack.

The UAV is assumed autonomous, meaning that it is not directly controlled by a human operator. However, some functionalities can be delegated to the infrastructure, that is, the UAV might depend on the help of other remote devices. In general, autonomy requires the acquisition and processing of signals from the surrounding environment to inform control. This transformation of the input signals to the output control has a rather broad meaning. The UAV can be simply programmed to follow a trajectory through a series of waypoints,

in such case the input signal are GPS coordinates, and the UAV determines its motion to reach the next waypoint from its current position. However, in many practical use-cases, an autonomous UAV may need to acquire and process complex signals, a task which may impose a significant burden to the battery-powered UAV in terms of energy expense. Furthermore, due to weight constraints UAVs often have weak on-board computing platforms, which may lead to a large time needed to execute compute-intense algorithms. Thus, offloading to compute-capable edge servers may lead to considerable energy consumption reduction as well as a faster capture to control pipeline for autonomy.

We provide in the following a short summary description of the class of infrastructure assistance problems we consider in this work. The UAV produces a series of bursts of data at the application layers. We define, then, the series of instants $\{\tau_i\}_{i=1,2,3,\ldots}$, where τ_i corresponds to the emission time of the i–th burst. The bursts are characterized by a variable set B_i that determines the amount of data generated in the i–th burst. We abstract the communication between the UAV and the access point – Wi-Fi access point or LTE eNodeB in the considered environment – through the transformation Φ, which maps the emission time τ_i and burst size B_i into the vectors $\mathbf{t}_i$=$[t_i(1), \ldots, t_i(N_i)]$ and $\boldsymbol{\omega}_i$=$[\omega_i(1), \ldots, \omega_i(N_i)]$. The vector $\mathbf{t}_i$ contains the delivery times of the N_i network packets associated with the burst, the element $\omega_i(n)$ of $\boldsymbol{\omega}_i$ is equal to 1 if the packet is delivered and 0 otherwise. For convenience, we set to ∞ the delivery time of a failed packet. We define as Δ_i the difference between the delivery time of the packet of burst i delivered the latest and the generation time τ_i.

Intuitively, the transformation Φ is a function of a number of variables describing the complex state of the network. Clearly, analytically modeling the interactions between channel physical, access, link and transport layer protocols implemented by all the active nodes, as well as their packet arrival processes, is an impossible task, and we rely on detailed simulations to characterize the transformation Φ associated with specific communication strategies and environmental conditions.

8.3.1 Network Environment

In urban network infrastructure, the most commonly used wireless technologies are WiFi and LTE. WiFi is traditionally operated in unlicensed frequency bands, whereas LTE primarily operates in licensed spectrum – although recent propositions extend its usage in the unlicensed spectrum as well. WiFi is a bidirectional communication technology specified in the IEEE 802.11 standard. In common infrastructure, WiFi stations (users) connect to an access point (AP) to communicate with other users or a remote server. The most used Medium Access Control (MAC) layer is Distributed Coordination Function (DCF), where the users contend for channel access using channel sensing and random backoff. Recent standards of WiFi support bitrate adaptation to improve the reliability of communications and Orthogonal Frequency Division Multiplexing (OFDM) modulation to increase capacity.

LTE is a cellular technology which defines an access network component called Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and a core network component called Evolved Packet Core (EPC). The LTE access network has separate uplink and downlink radio bands, where resource allocation is controlled by the base station (eNodeB). The radio resources in LTE is allocated at the granularity of individual Physical Resource Block (PRB) which is the minimum unit that can be assigned to one user. Each PRB corresponds to one slot in the time domain and contains 12 subcarriers of 15 kHz each, thus occupying 180 KHz in the frequency domain. The eNodeB scheduler allocates a subset of PRBs to each user for each subframe based on a scheduling policy. In Uplink, the PRBs allocated to a specific user should be contiguous, whereas in downlink the eNodeB can allocate non-contiguous PRBs to a user. Additionally, as the transmission power of the User Equipment (UE) is lower compared to that of the eNodeB, uplink is performed using Single Carrier FDMA (SC-FDMA) – also known as DFT pre-coded OFDMA – which provides a better peak-to-average power ratio. The instantaneous data transmission rate over the LTE physical channel is determined by the the transport block size (TBS) which is a function of the number of PRBs allocated

to the UE and the Modulation and Coding Scheme (MCS) index. The transmitter receives the Channel Quality Index (CQI) information through a control channel and, based on the CQI level, it selects the appropriate MCS index.

In the urban infrastructure, both WiFi and LTE networks are available to assist the UAV to enable the autonomous mission. However, the different characteristics of the protocol of the two technologies introduces different challenges in terms of protocol overhead, reliability, handling mobility, congestion and interference.

8.3.2 Infrastructure Assistance

Within this general environment, we consider the two specific infrastructure-assisted models described in the following.

Remote Navigation Assistance: In a scenario where multiple UAVs share the same space, a remote unit can assist their navigation to avoid collisions, for instance by creating safe corridors for each UAV. To this aim, the UAV periodically transmits telemetry variables – GPS coordinates, altitude, speed and battery level and other information if enabled on the UAV – to the remote controller. The telemetry information are short messages usually contained within the maximum transmission unit (MTU) size of one network packet. The packets are usually sent at some update interval based on the polling request set by the GCS to fetch the telemetry information. Let's denote as $\mathbf{s}(t)=(p(t), v(t), b(t))$ the state of the UAV at time t, where $p(t)$, $v(t)$ and $b(t)$ are position, speed, and battery level, respectively. In our model, the emission of telemetry variables corresponds to a small data burst, that is, burst i contains the state $\mathbf{s}(\tau_i)$. Based on the received telemetry, the remote controller maintains a state estimate $\tilde{\mathbf{s}}(t)$. Clearly, as the UAV emits telemetry only at the time instants $\{\tau_i\}_{i=1,2,3,\ldots}$, even in an idealized case where $\Delta_i=0$ and $\omega_i=1$, $\forall i$, there is a mismatch between the actual and estimated state of the UAV due to the evolution of the state in between

telemetry emission. The network transformation might increase the estimation error by delaying telemetry delivery and erasing some updates due to packet failure. Intuition may suggest that more frequent updates would decrease the estimation error. However, frequent updates may increase network congestion and buffer overload at the source.

Computing Task Offloading: As discussed earlier, the infrastructure can provide a deeper degree of assistance by taking over the execution of complex processing tasks. For instance, a UAV may capture pictures in pre-set locations to detect events of interest (*e.g.*, a car accident or suspicious human activities). In case of a positive detection, the UAV may be programmed to autonomously alter his course and acquire more information in that specific area. In a more extreme example of autonomy, the UAV may be closely pursuing a specific object based on sensor feed. In this case, as the UAV uses the outcome of processing to control fine-grain motion, the frequency of offloading is much higher compared to the previous example, and the requirements on delay and delay variations much more stringent. Clearly, the emission of processing tasks corresponds to the emission of a burst. The level of autonomy and the nature of the task (*e.g.*, object detection on a video stream) determines the burst size distribution, as well as the delay and reliability requirements on the task delivery. In a typical use-case scenario, the on-board camera captures video with 720p HD resolution for video streaming, that requires a throughput of at least 1.8 Mbps. Depending on the computation requirement, one can vary the frame rate or resolution and hence modify the task size. Usually, for tasks like image classification or object detection an entire image frame is sent as a burst of data to the server. For example, if images of average frame size $150KB$ are captured with 10 frames per second, and one application packet size is about 1500 bytes (including all headers) then a burst of $(150000/1500) = 100$ packets will be sent over network in every 100 milliseconds.

The above case-study scenarios generate a broad spectrum of characteristics of infrastructure assistance in terms of both generated traffic – frequency and size of bursts – and requirements.

Our simulation study attempts to bridge the UAV and communication/networking domains to provide a comprehensive discussion on the feasibility of infrastructure assistance to UAV operations.

8.4 Simulation Environment

We briefly summarize the general settings used to obtain the results presented herein.

8.4.1 Urban Environment

Based on the discussion presented earlier, it is clear that infrastructure assistance is a key enabler of autonomous UAV technologies. However, the UAVs and the infrastructure are necessarily connected through volatile wireless links. Especially in urban environments, parameters such as network load, signal propagation and mobility may degrade the ability of these links to reliably support assistance. In the following, we briefly describe the most salient aspects we explore in our simulations.

Propagation

In urban environments, high buildings often induce Non-Line of Sight (NLoS) signal propagation, where penetration loss and reflection through on building walls can cause highly varying attenuation. The dimension and location of the buildings, the thickness and materials of the walls also contribute to determine the overall amplitude of the received signals. The reliability of infrastructure assistance can be greatly impaired by these characteristics.

To model these properties, we adopt a pathloss model for urban environments built as the combination of the standard ITU R1411 pathloss model [107] and other variable components

function of salient parameters. Specifically, the path loss is defined as :

$$L = L_b + L_{ew} + G_h, \tag{8.1}$$

where L_{ew} is the loss through the external walls and G_h is the gain due to the altitude of the device. The loss L_b is the basic pathloss function

$$L_b = \begin{cases} L_{los}, & \text{if } Line\ of\ sight \\ L_{nlos}, & \text{otherwise} \end{cases} \tag{8.2}$$

In line of sight, the pathloss term is defined as:

$$L_{los} = \mid 20log\ (\ \frac{\lambda^2}{8\pi h_{bs} h_{uav}}) \mid + C, \tag{8.3}$$

where h_{bs} and h_{uav} are the altitudes of the base station and UAV respectively, λ is the wavelength and C is a variable whose upper and lower bound are a function of distance, wavelength and altitude of the devices. In LoS, the pathloss term is defined as:

$$L_{nlos} = L_{fs} + L_{df}, \tag{8.4}$$

where L_{fs} is the free space pathloss and L_{df} is the total pathloss due to diffraction, including that generated by the roof top and from the rows of the buildings. The value of L_{df} depends on the frequency of the signal, distance of transmitter and receiver, and altitude of the devices. Note that urban pathloss not only restricts the communication range, but also has large impact on resource allocation and modulation schemes in the communication. In our feasibility study, we consider non-LoS pathloss model.

Mobility

The maximum speed of UAVs can be of the order of 100 mph and the mobility patterns depend on several factors including the mission objective and environment. Unlike MANET, the UAVs move in the network in a 3D space leading to an increased range of dynamics, and the high speed leads to a faster change in terms of channel gain. In infrastructure-based UAV network, UAV may go out of coverage of the current base station or access point. However, most of the infrastructures provide handover mechanisms for continued connectivity.

Exogenous Traffic

In case of infrastructure-assisted UAV networks, the mission-based application data stream is affected by the contention of data streams generated by users using the same access point or base station. The infrastructure can provide flow control and congestion control, and attempt to achieve fairness in resource allocation among coexisting applications and communicating devices.

Note that interference can be generated by coexisting wireless networks or the presence of local Device-to-Device (D2D) communications using the same frequency band. The infrastructure can implement a centralized or distributed coordination in terms of transmission scheduling and power control to avoid or mitigate the impact of interference. Especially when heterogeneous network technologies share same spectrum causing interference, the different technologies can cooperate through the core backbone network to improve overall performance.

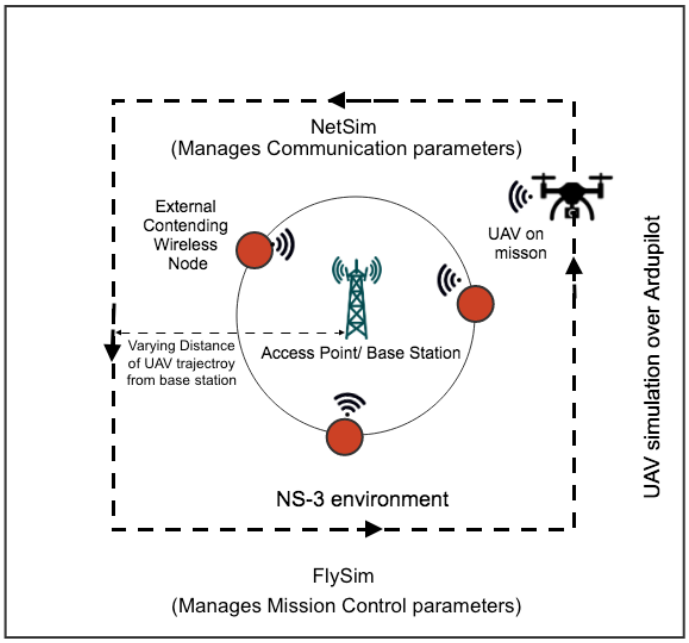


Figure 8.2: Simulation setup: the UAV follows a predefined trajectory, whereas external nodes are fixed and disposed in a circle centered on the AP or eNodeB.

Network Technology

Technologies and protocols are often designed under certain assumptions and tailored to specific scenarios. The technologies in urban environments have different specifications and approaches to data transmission and channel sharing, e.g., WiFi employs a distributed algorithm for contention based resource allocation, whereas in LTE resource allocation is controlled using a centralized policy by the base station. The technologies also differ in terms of retransmission strategy, error correction, interference control, security and many other aspects which advocates to choose appropriate technology and protocol in different application scenarios and different network conditions.

8.5 Results and Discussion

8.5.1 Simulation Setup

We simulate the environment and applications described earlier using the integrated UAV-network simulator FlyNetSim. The simulated UAV performs a predefined mission, which corresponds to a navigation plan in an urban environment. The wireless environment is created using ns-3, the network simulator component of FlyNetSim. The mobility of the vehicle is updated in the network simulator in real time and network parameters are varied to evaluate the performance in a range of scenarios. A realistic simulation of motion and control and control of the UAV is obtained using ArduPilot. The simulated UAV uses simulated sensors to provide telemetry information to the GCS over a simulated network. In the second application, the UAV generates tasks/packet bursts with different statistics and uses simulated WiFi and LTE networks to transport them to an edge server.

Ground WiFi/LTE nodes are added with uniform disc position allocation close to the GCS node. These nodes produce data streams directed to the same access point or eNodeB. The number of external nodes, the datarate and the packet size of the ground users are tunable parameters. The pathloss and shadowing parameters are set based on the ItuR1411 Propagation Loss Model. Motion, shadowing and exogenous traffic all affect the transformation induced by the network on the packet stream from the UAV. The parameters used in the simulation are summarized in table 8.1. The UAV moves in a rectangular trajectory around the AP or base station as shown in Figure 8.2.

Parameters	Value
UAV Mobility	Constant Speed
WiFi Standard	IEEE 802.11a
WiFi Bandwidth	20 MHz
Propagation loss Model	ItuR1411 Propagation Loss
Propagation Delay Model	Constant Speed
LTE EARFCN	18000
LTE Bandwidth	20 MHz
LTE RLC Mode	Acknowledgement (RLC AM)
LTE downlink MAC Scheduler	Proportional Fair
LTE uplink MAC Scheduler	Round Robin
TCP Congestion Control	New Reno

Table 8.1: Experiment Parameters used in the simulations.

8.5.2 Remote Navigation Assistance

Figure 8.3 shows an example of temporal evolution of error in position estimation while the UAV performs a mission over the predefined trajectory. In this experiment, communication is over WiFi, and the different lines correspond to a different number of nodes contending the shared channel resource with the UAV. The contending nodes are placed with uniform disc position allocation in close distance from the AP/base station and transmits periodic bursts of packets of size 800 bytes with uniform periodicity based on the traffic rate; e.g., for traffic rate of 6 Mbps, the packets are transmitted at interval of $(800 * 8)/6 = 1067\mu s$. It can be seen how not only the mean of estimation error grows as the number of contending nodes increases, but also the variance. The line corresponding to 2 nodes has few sections where the error doubles, probably caused by TCP timeouts, but otherwise shows a considerable stability. Note that the error has periodic low spikes, corresponding to periods of time where the UAV is stationary. A higher number of nodes contending the wireless resource further increase average and variations of delay, with extended sections of the trace where the error significantly increases with respect to the minimum. This latter effect maps to a general inability of a remote node to maintain a tight control over UAV trajectories.

We measure the average performance of telemetry transmission for navigation and compu-

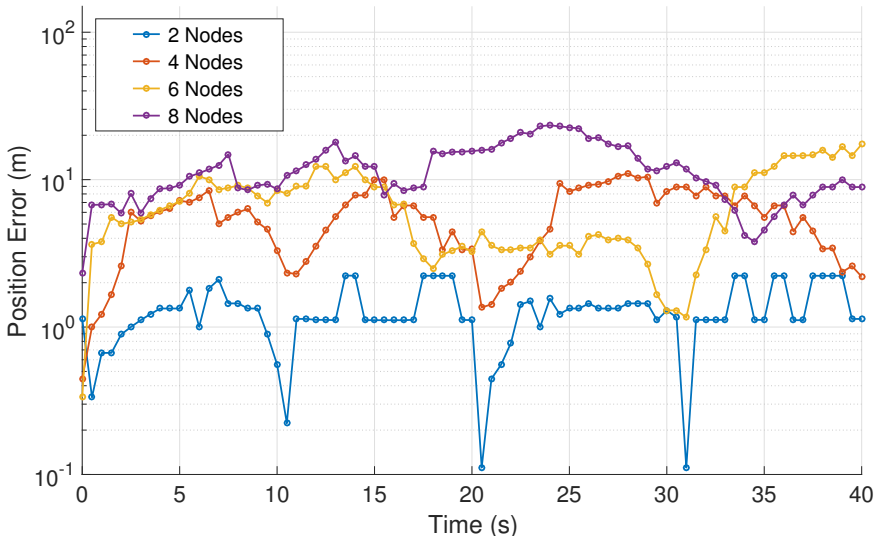
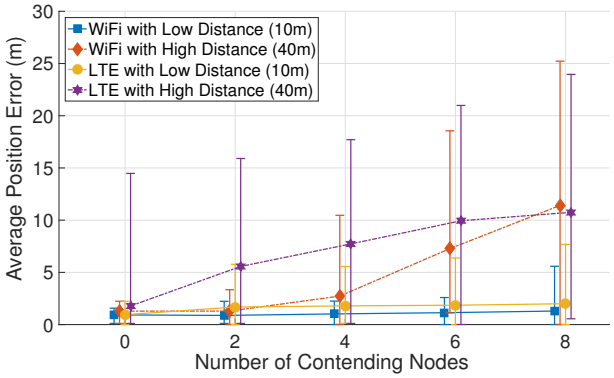


Figure 8.3: Temporal evolution of position error based on the telemetry emitted by the UAV over WiFi with TCP in an urban environment with buildings. Each external node transmits traffic with rate 6 Mbps. The UAV navigates on a rectangular trajectory where the access point is at the center.

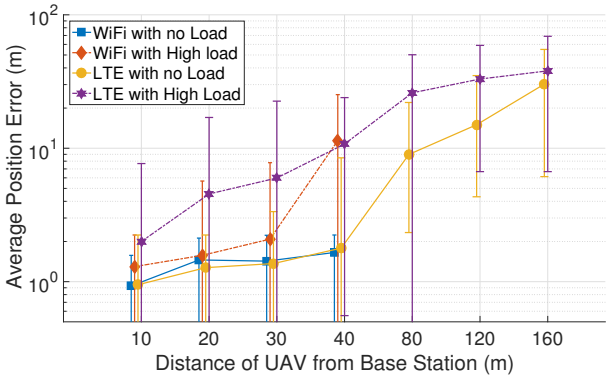
tation task offloading in various scenarios. For the external traffic, we define two extreme regimes - *no load* conditions, where no wireless node other than the UAV is present, and *high load* conditions, where 8 contending nodes are present generating 6 Mbps in the application layer. We also define *Low Distance* and *High Distance* scenarios, where the distance between the UAV and the access point or base station is 10 and 40 m, respectively.

Figure 8.4a compares the average position error *High* and *Low Distance* conditions over WiFi and LTE networks as a function of the number of ground nodes. It can be seen how both networks maintain a low error as the number of nodes grow, with WiFi being a slightly better option. At 40 m, WiFi has an manifestly lower error compared to LTE in mild contention environments, but presents a sharp degradation as the number of ground nodes grows. This is due to the inefficiency of DCF and random access in high-load conditions compared to the LTE more controlled MAC.

Figure 8.4b shows the same metric over WiFi and LTE in *High* and *No Load* conditions as a function of distance. Note that 40 m is the disconnection limit for WiFi, whereas LTE has a



(a) Average position error and variation range for different communication technologies and protocols with respect to varying number of contending nodes. Each node is transmitting 6 Mbps data traffic.



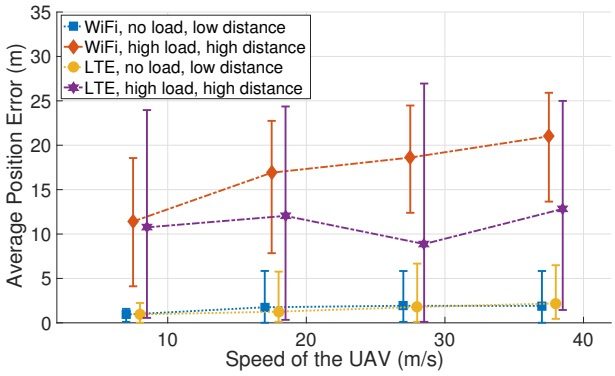
(b) Average position error and variance as a function of distance between the UAV and the base station for different communication technologies. The measurements are taken in no load and high load (8 nodes 6 Mbps each) conditions.

Figure 8.4

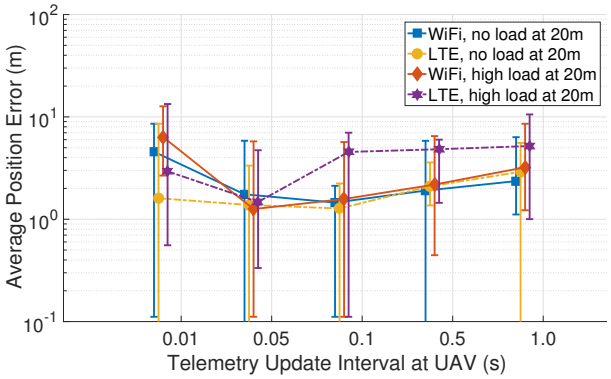
much extended range, although at the price of a large position error. The effect of distance and load is apparent in both technologies. However, it can be seen how in the absence of traffic from other nodes the two options are essentially equivalent until disconnection. Conversely, in *High Load* conditions the difference is marked: WiFi has a much lower error at moderate distance, and then sharply degrades as the UAV approaches the maximum range.

We also measure the impact of speed on the position error. The plot is in *High Distance* regime. Intuitively, a higher speed of the vehicle would result in a larger error simply due to the fact that the UAV would have moved farther since a packet containing the last update was received. Figure 8.5a shows the effect of increasing speed of UAV in *High* and *No Load* conditions. As expected, the absolute position error increases with the vehicle speed. The impact of load is also manifest: the higher inter-packet delay maps to a faster error increase.

We also measure how the frequency of updates from the the UAV affects the absolute value of the position estimation error. Figure 8.5b shows the variation at distance (20m). Intuitively, a low update frequency generates a small network load, but also allows the UAV to travel



(a) Variation of Average position error with the speed of the UAV in different network load conditions over LTE and WiFi.



(b) Variation of Average position error with different update frequency at the UAV for telemetry over WiFi and LTE.

Figure 8.5

farther in between updates. Conversely, frequent updates means a smaller error in idealized conditions, but impose a larger load to the network. This trend is shown in the plot, where the error has a minimum which depends on the technology and external load. Note that the effect of the additional load introduced by frequent updates is more pronounced in LTE *High Load*, due to the smaller maximum throughput of the network in those conditions.

8.5.3 Computing Task Offloading

We now focus on evaluation of the performance when the UAV is transmitting burst of packets with certain characteristics.

Figure 8.6 shows the variation of task delay as a function of distance in a scenario where the tasks correspond to small 50 KB data are transmitted from the UAV every second. Interestingly, it can be observed that in this case load conditions have a small impact on WiFi, whereas LTE suffers a larger number of nodes using the same channel. LTE clearly outperforms WiFi in *No Load* conditions unless the UAVs is very close to the access point. Conversely, WiFi has a smaller error compared to LTE in *High Load* conditions up to 30 m distance, where the smaller range of WiFi penalizes this choice. These results are strongly

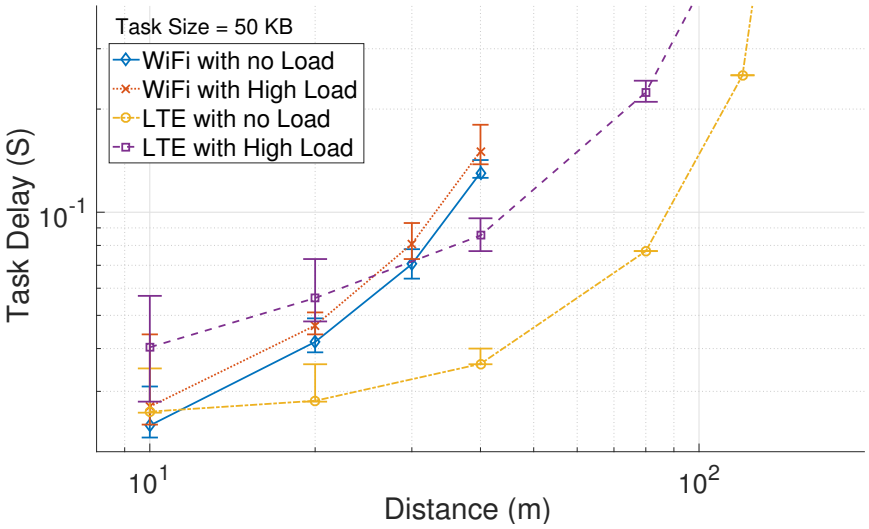


Figure 8.6: Task delay over WiFi and LTE with varying distance of the UAV from the Base Station for task of burst size 50 KB every second. The measurement are taken in two different external traffic regimes of no load and high load.

influenced by the small size of the task, which makes the corresponding packets go through transmissions in the WiFi MAC. In LTE, the round robin allocation of resources increases the delay in the presence of other users.

Figure 8.7 shows the same plot where tasks are of size 200 KB. It can be observed a general shift of all the delays, with WiFi being the most penalized by the size increase. One of the reason for LTE incurring a less perceivable degradation with respect to WiFi is the RLC buffer, that reduces the retransmission at the TCP layer [124] compared to WiFi where the large burst size can cause more back-offs. However, when the network is congested, resource allocation still penalizes LTE in uplink as round robin is used.

We give a more clear view of this effect in Figure 8.8a, where we show delay as a function of task size in *No Load* conditions. WiFi suffers a larger delay at high distance, but provides better performance compared to LTE in the short range. It can be seen how WiFi has a steeper delay increase as the task size increases.

Finally, Figure 8.8b shows that in presence of external traffic in the network, WiFi has a

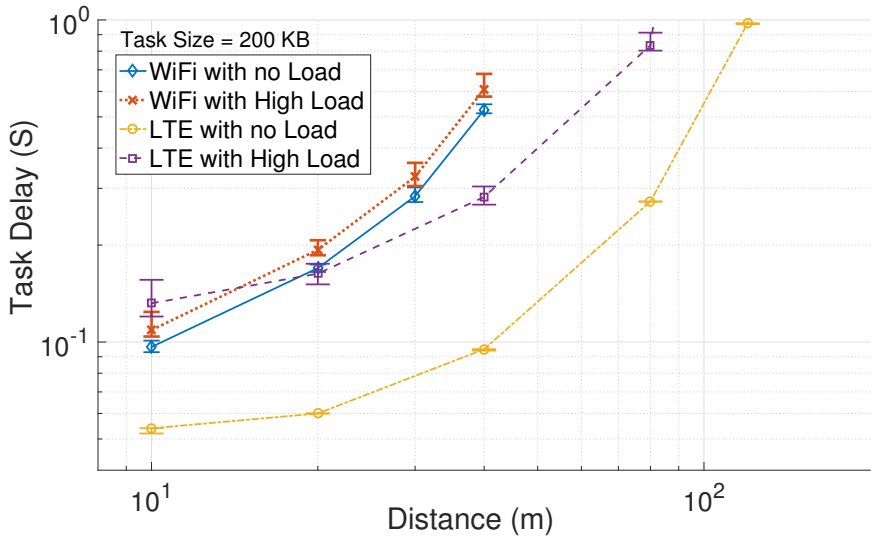
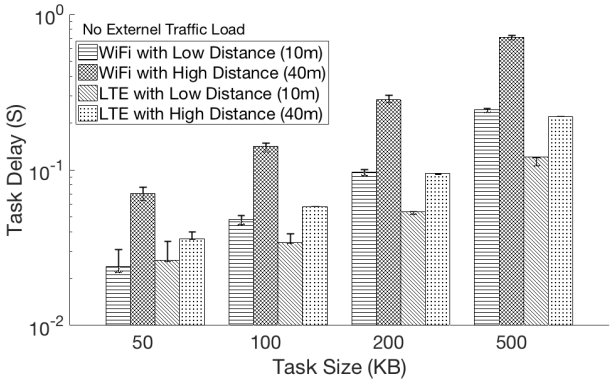
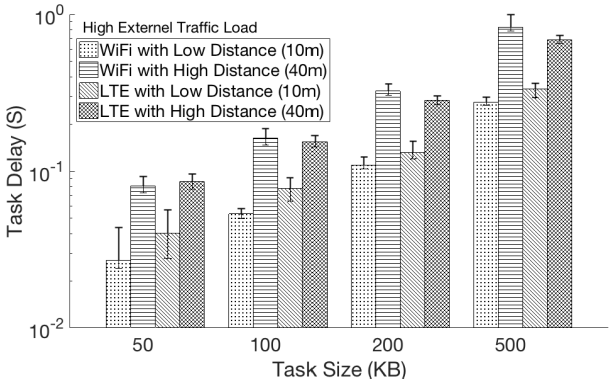


Figure 8.7: Task delay over WiFi and LTE with varying distance of the UAV from the Base Station for task of burst size 200 KB every second. The measurement are taken in two different external traffic regimes of no load and high load



(a) Task delay over WiFi and LTE with varying task sizes for low (10m) and high (40m) distance of the UAV from the Base Station. The measurements are taken in absence of any external traffic load.



(b) Task delay over WiFi and LTE with varying task sizes for low and high distance of the UAV from the Base Station. The measurement are taken in in presence of external traffic load of 3 nodes transmitting traffic of 1 Mbps each.

Figure 8.8

smaller delay compared to LTE in closer range for all task sizes. This is due to the fact that the task is transmitted over LTE uplink which schedules in round robin all the UE nodes data. Also, it adds additional delays every time it needs to seek transmission opportunity from the eNodeB for a new chunk of data from any given UE. However, in longer distance, WiFi deteriorates fast and slightly worsen the performance compared to LTE.

8.5.4 Discussion

Based on the results from two use case scenarios of infrastructure-assisted UAV, we can see that the choice of network depends on the network conditions, distance from the access point or base station, as well as the class of application that UAV is serving. In remote navigation assistance, the telemetry data are more suitable to be transmitted over WiFi when the UAV is close to the AP. In applications such as task offloading, the size of the task has a great impact on the network to be used. When large data bursts are to be transmitted efficiently to the edge with low latency, the LTE network provides best performance in low-load conditions. However, if the network is congested, connecting the UAV over WiFi is still advantageous at short distance from the AP, whereas LTE is the best option. We conclude that in order to achieve efficient network infrastructure-assistance, the UAV should be multi-homed, that is, it should have both WiFi and LTE interfaces and use a context and application aware policy to determine which network should be used.

8.6 Conclusions

The main objective of this work is to provide a comprehensive evaluation of communication strategies for infrastructure assistance to the operations of autonomous UAVs. Based on detailed UAV-network simulations, we focus our attention on remote navigation assistance and offloading of processing tasks to edge servers. Our results, obtained using the recently proposed UAV-network simulator FlyNetSim, indicate the need for the UAVs to be equipped with multiple network interface and be able to switch from one to another during missions.

Chapter 9

Building a Robust Airborne System with Real-world UAVs

9.1 Introduction

On-board resources available to commercial Unmanned Aerial Vehicles (UAV) are inherently limited by the airborne nature of these devices. Such constraints affect key sub-systems, such as sensors and computing platforms, which are instrumental for prolonged and autonomous operations. A relevant trend in mobile devices, whose application has been recently extended to include UAVs, is to use resources from the Internet of Things (IoT) and edge infrastructure surrounding the UAVs to enhance their performance [64, 70]. In particular: *a)* wireless cellular networks can be used to extend the communication range of the UAVs and connect them to other edge devices or the internet core [103]; *b)* data streams from ground sensors or devices can supplement those from on-board sensors to extend the information available to the UAVs [87]; and *c)* signal processing tasks can be offloaded to compute-capable devices residing at the network edge, that is, edge servers [141].

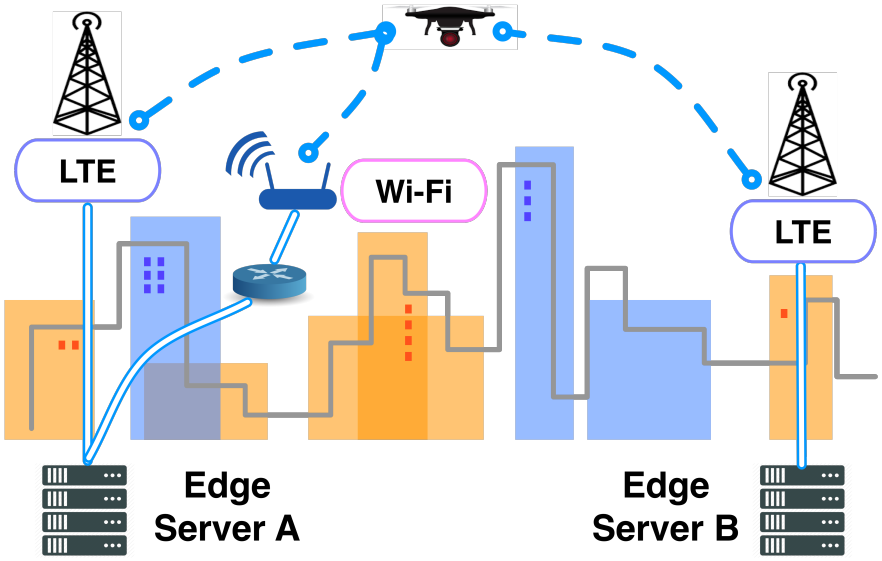


Figure 9.1: The scenario considered : an autonomous UAV leverages the surrounding IoT infrastructure to improve mission performance. Specifically, we focus on offloading computing tasks to edge servers through wireless links.

Although most of the discussion provided in this work applies to all the three cases listed above, herein we focus on edge offloading – case *c)*, often referred to edge computing [87] – due to its potential to enable a high degree of operational autonomy to those highly constrained devices. The advantages of edge computing applied to UAV systems are rather intuitive: a powerful machine with an unlimited energy supply connected to a UAV through a one hop wireless link can effectively run complex algorithms beyond the reach of on-board UAV computing platforms and deliver the outcome to the UAV, thus enabling advanced – autonomous – learning, control and planning. The edge server can run simpler algorithms in a shorter time, thus improving the reaction time of the UAV to external stimuli. Finally, we remark that continuous computation requires a substantial amount of energy, a precious resource that should be parsimoniously used to avoid degradation of mission lifetime. By taking over compute-intense processes, the edge server can significantly reduce energy consumption of the UAV.

Based on the above discussion, edge computing, and in general infrastructure assistance to UAV systems, appears to be an extremely promising component of future UAV systems

(see Fig. 9.1). However, there are several technical challenges to overcome to fruitfully and reliably apply this paradigm to a mission-critical system such as autonomous UAVs. A crucial aspect of the infrastructure assisting the UAVs is that it is shared by a multitude of devices and services. This, together with the characteristics of signal propagation, creates an extremely dynamic environment, where the time needed to transfer data to the edge server, or the time to complete the processing task are highly variable, and governed by extremely complex temporal and spatial random processes. As a result, blindly trusting infrastructure assistance may lead to severe performance degradation.

We contend that to fully harness the benefits of infrastructure assistance while preserving reliable flight and mission control the notion of "autonomy" needs to be extended from pure mission control to include an advanced layer of intelligence making decisions on how information is handled within the complex UAV-infrastructure system. We refer to this layer as "information autonomy". We remark that information autonomy could boost the performance of many distributed, and constrained, mission-critical systems. We make our discussion specific to UAVs due to their extreme characteristics in terms of limitations, mobility and complexity of operations.

Herein, we provide a first description of an architecture realizing information autonomy, and a detailed discussion on the many challenges present in the definition of key modules such as state acquisition/tracking, prediction and decision making. The discussion is supported by illustrative results from detailed simulation and real-world experiences and implementations.

9.2 Challenges in Real-world Experiments on UAV Systems

Figure 9.1 illustrates the autonomous infrastructure-assisted UAV system at the center of this research study. In the edge computing scenario, the UAV is connected to one or more edge servers, positioned at the network edge for handling resource-intensive computations, through available wireless access networks, such as Long-Term Evolution (LTE) or Wi-Fi. Edge servers are preferred over cloud infrastructure due to their low latency.

The edge server would take over the analysis, and possibly the control, operations of the profiling-analysis-control pipeline typically seen in UAV applications. In such Edge-based computation pipeline, the input generated by the sensor and the output from the modules at the edge server, needs to be wirelessly transferred to and from the edge server. Importantly, as compute-intense tasks often require a large amount of energy, offloading typically reduces energy consumption at the UAV, even when considering the additional energy needed to transmit the data to the edge server.

For a successful completion of a mission, all the hardware components of UAV have to continuously generate new knowledge for the control algorithm in a dependable and energy-efficient manner. Unfortunately, it is difficult to estimate the resources needed for sensing, actuation, communication, and control before the mission (run-time) since the stability of the drone is impacted by various environmental, system, and network components. In the rest of this section, we present the lessons learned from our real-world experiments.

9.2.1 Run-time Uncertainties of Edge-assisted UAV Applications

We have carried out a set of experiments involving drones and edge devices following the setup as shown in Figure 9.2 to understand the performance and the resource constraints of edge-assisted UAV applications. The UAV is equipped with small on-board computer connected with WiFi dongle and USRP B200mini for LTE interface. The edge server is realized on a Laptop with high computational capability, that is also connected with WiFi and USRP B210 for LTE interfaces. We tested a UAV mission of target tracking based on image classification.

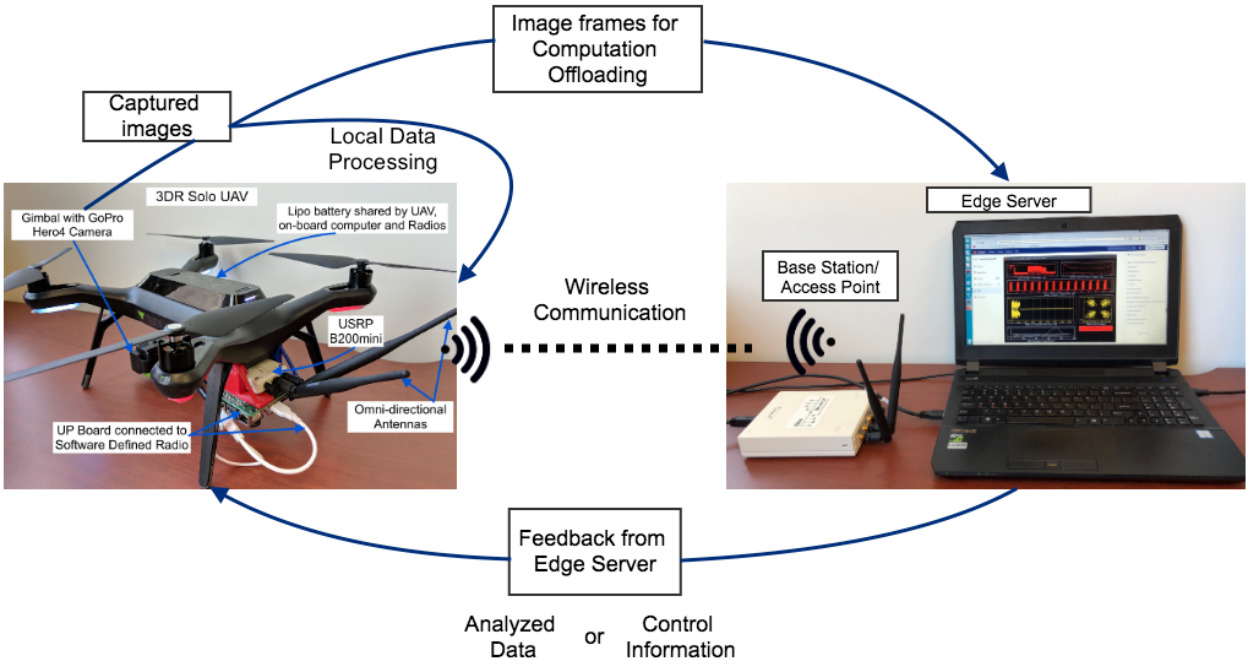


Figure 9.2: Experimental setup for UAVs with on-board computing and communication equipment.

From our real-world experiments conducted in the outdoor, we have identified the following uncertainties in managing the edge-assisted UAV applications:

- $\mathcal{U}$**1. Environmental Uncertainty** Weather pattern such as wind speed, temperature, and humidity influences the control algorithm since the drone needs to stabilize itself

under harsh conditions by controlling its engines. Due to the unpredictable nature of the weather, it is hard to allocate resource budget for control operations prior to the mission.

- $\mathcal{U}$**2. System Uncertainty** For surveillance or other edge-assisted UAV applications, performance depends on the computation capacity of the drone, and the energy budget available for various operations including computation, communication, sensing, and storage (e.g., buffer).

- $\mathcal{U}$**3. Network Uncertainty** Wireless communication is susceptible to interference, hence the edge-assisted UAV applications relying on remote edge servers have to carefully manage the network resources including the bandwidth capacity of the wireless link and the energy budget needed to operate the radio hardware.

The above challenges highlight the need to introduce "Information Autonomy" to react and adapt to the system, network, and environmental uncertainties. Application goals have to be updated dynamically during the mission time to maximize the mission duration. In the next section, we formulate problem goals with respect to the application scenario.

9.3 Autonomous Mission-oriented Applications over UAV

We consider an UAV autonomously operating to fulfill a mission. Here, we do not specifically define the mission, but, instead, assume that autonomous operations necessitates sensor input to be transformed into short- and/or long-term decisions. For instance, video input from on-board cameras can be processed to determine navigation, or to plan the mission. We further assume that our objective is to assure the mission completion, therefore maximizing mission time, while maintaining quality of operation, as discussed below.

Figure 9.6 depicts a basic diagram of the modules involved in the transformation from input vector $\mathbf{u}_t$ to the output vector $\mathbf{y}_{t+\Delta}$ given a context or state $\boldsymbol{\phi}_t$, where t is the time at which the sensor input is acquired. The input is first processed in the analysis module to extract relevant features. For instance, in a video-based navigation application, the input is a video frame, and the output of the analysis module, generated by an object detection algorithm, is a set of labeled bounding boxes. The features produced by the analysis module are sent to the control module, which generates the final output $\mathbf{y}_{t+\Delta}$. In this particular example, the output are motion and navigational commands.

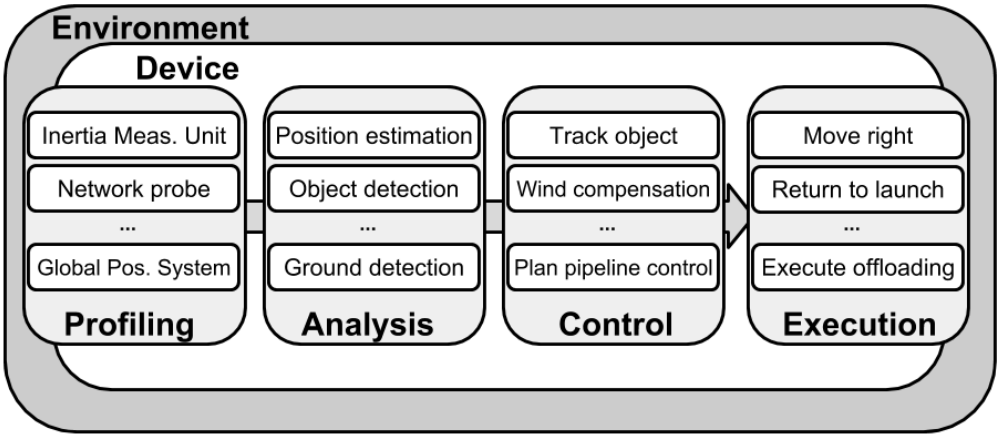


Figure 9.3: Profiling-Analysis-Control Pipeline for Mission Autonomy.

Formally, we denote the transformation using the following:

$$\mathbf{y}_{t+\Delta} = f(\mathbf{u}_t, \boldsymbol{\phi}_t). \tag{9.1}$$

Note that the output is generated at time $t+\Delta$, where Δ is a random variable corresponding to the time needed for the transformation. We refer to Δ as the *capture-to-output* delay.

If the function f is complex, the delay Δ might be large, thus increasing the reaction time of the UAV to input stimuli and possibly degrading mission performance. Moreover, compute intense transformations may require a large amount of energy E to be completed. *Our objective, thus, is to to use infrastructure-assistance to reduce as much as possible* Δ*, while*

also minimize energy consumption.

In the edge-based pipeline, the capture-to-output time, Δ, is a function of many variables describing the environment. Let's decompose the delay of the edge-based pipeline Δ^e as follows:

$$\Delta^e = \Delta^e_{\text{data}} + \Delta^e_{\text{computing}} + \Delta^e_{\text{output}}, \tag{9.2}$$

where Δ^e_{data}, $\Delta^e_{\text{computing}}$, and Δ^e_{output} are the delay to transport the data to the edge server, the processing time at the edge server and the time to transport the output back to the UAV, respectively.

Intuitively, the components associated with the wireless transfer of data, *i.e.*, Δ^e_{data} and Δ^e_{output}, are a function of the used technology (*e.g.*, LTE and Wi-Fi) channel characteristics (UAV-base station distance, fading and shadowing), and network load. The processing component of the delay, that is, $\Delta^e_{\text{computing}}$, is highly dependent on the server load. Importantly, channel gain, and network and server load are highly variable, and have complex spatio-temporal distributions that play an important role in building information autonomy.

9.3.1 Need for Information Autonomy

To study the dynamics of Edge-assisted UAV applications, we used the integrated UAV-network simulator – FlyNetsim [38]. In Figure 9.4 we provide temporal traces generated using the simulator. The traces show Δ_{data} over time in our object detection setup, where the UAV transfers a picture to the edge server using Wi-Fi as a function of the number of competing wireless nodes and their transmission rates. It can be observed that as the traffic generated by each node increases (see Figure 3a), not only the average delay increases, but

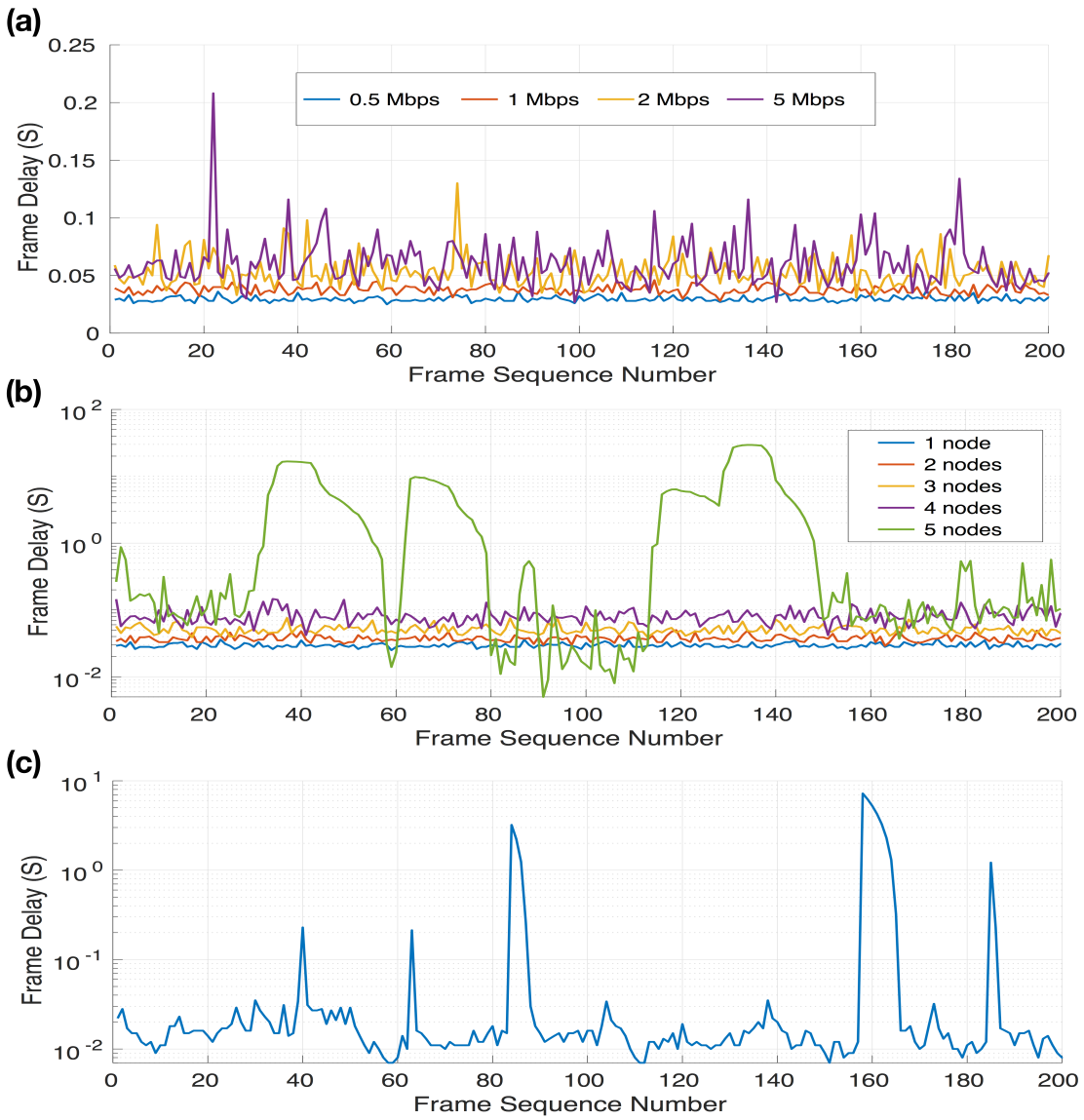


Figure 9.4: Frame delay w.r.t. varying network loads & mobility.

the temporal variations across subsequent frames become more apparent. This is due to the complex interactions between the streams induced by the transmission and transport layer protocols, which manifest especially as the sum traffic approaches the maximum link capacity. Similarly, if we increase the number of competing nodes (see Figure 3b), we will reach a point in which packet failures or timeouts will induce large variations. The bottom plot (see Figure 3c) in the picture shows the abrupt variations in the delay Δ_{data} induced by random motion of the UAV when adaptive rate is used. Importantly, the underlying

conditions, channel gain, number of nodes, task load, vary over time and space, presenting different trends for different context.

From the considerations, it is clear that offloading may not be consistently beneficial, at least from the point of view of capture-to-output delay. Moreover, the dynamics of capture-to-output delay highly depend on the characteristics of the surrounding environment. A simple reasoning and modeling on the key factors – network rate and server load – can be found in [58].

9.3.2 Related Work:

Information Autonomy problem presented in this study is a form of Multi-Criteria Decision Making (MCDM) problem [92]. The configuration space for the Information Autonomy consists of multiple parameters that control the wireless performance, processor efficiency, energy consumption, and response time of the application with the goal of maximizing the mission time. Existing self-adaptation approaches such as Model At Run-time [202, 88] and control theory methods [175] can be applied to achieve information autonomy for Edge-assisted UAV systems. However, such approaches are not optimal and may have to be carefully tuned to reduce "capture-to-control" delay.

Weyns *et al.* [202] presents, MARTAS, an architectural based adaptation approach for industrial IoT wireless networks. MARTAS uses Models At Run with statistical techniques to reconfigure the network parameters of a multi-hop wireless network protocol. Paucal *et al.* [88] introduces a run-time adaptation technique based on Dynamic Decision Networks (DDN) and Analytic Hierarchical Process (AHP). The proposed technique enables the application to reconfigure priorities based on the data collected by the system during run-time, which makes the solution effective for applications that has limited knowledge at the design time.

Shetsov *et al.* [175] surveys the control theory approaches for self adaption and shows how linear control models are applied to self-adaptive software systems. Filieri *et al.* [80] presents an example of how control theory can be used to create a self-adaptive controller for software systems. Applications and systems that exhibit high level of dynamism such as the Infrastructure Assisted UAVs require solutions that converges quickly to an optimal set point when making decisions through the sense-analysis-actuation loop. The longer responsive time and the inaccurate control setting leads to high level instability for highly dynamic systems. However, the control theory approaches are still promising for the problem discussed in this work.

9.4 Information Autonomy

Our simulation results in Section 9.3.1 show that a predefined offloading strategy would fail to harness the performance boost granted by infrastructure-assistance, or would possibly even degrade mission performance. We contend that due to the complexity of the involved dynamics, UAVs will need to implement a form of autonomy encompassing this relevant aspect of the system. We refer to this component of autonomy as *information autonomy*. Recall that the mission of information autonomy is to minimize the capture-to-output time Δ by selecting the best processing pipeline, either local or distributed over the infrastructure. As illustrated in Fig. 9.1, the UAV may have multiple options in terms of network and edge server.

Structurally, the pipeline to realize information autonomy is analogous to that shown in Fig. 9.6. However, the modules need to be specialized to the rather difficult problem of managing information in the composite UAV-infrastructure system.

9.4.1 Profiling

In most autonomous systems, this part of the pipeline is composed of sensors directly collecting signals from the environment. In the context of information autonomy, the Profiling module is assigned the difficult task of extracting information, a signal, from the infrastructure enabling the estimation and/or prediction of the state of entire communication-processing sections of the infrastructure. We identify two main strategies, which we refer to as inquiry and probing.

Inquiry: The sensing module sends direct inquiries to the infrastructure. For instance, the module can obtain from the network the expected transmission rate, and the number of current tasks – being executed or queued – from the edge server. This method presents two main disadvantages. First, the infrastructure would need to implement protocols to accept requests and compose replies. Additionally, in many network technologies the use of a channel would induce complex interactions with other active data streams, altering the available resource.

Probing: A simpler and possibly effective option is to introduce forms of probing, where the sensing module would "test" available pipelines by issuing tasks over the resource. Probing does not require any modification to the infrastructure, but incurs a drawback: the probe will use network and server resources, and for this reason probe size has to be finely tuned. This approach allows an objective observation of the state of the pipelines, but imposes a considerable burden to the infrastructure.[1]

A possible strategy to reduce the burden on the system is to reduce the size or frequency of the tasks sent within the probes. However, a reduced size or frequency of the probes may reduce the amount of information gathered on the state of the pipeline, as a small probe

[1]Note that if the full task is sent over multiple – independent – pipelines, the UAV can use the output generated in the smallest time, thus reducing the impact of unexpected variations through diversity.

may “traverse” the system unaltered, whereas a full task may induce complex effects.

9.4.2 Analysis

The Analysis module transforms the output of the Profiling module – for instance a vector of delays obtained using probes – into features of the sensed pipeline state. Intuitively, as the objective of information autonomy is to select the best pipeline, the features should allow prediction of future performance of available pipelines. Taking as example the traces shown in Section 9.3.1, the Analysis module could attempt to predict future delays or extract properties of their distribution to inform pipeline selection.

A major problem in this part of the information autonomy is the temporal scale of analysis and prediction. The module could extract a long-term state of the pipeline, corresponding for instance to the number of active nodes and their traffic in our illustrative example. Tracking such state would roughly correspond to mapping samples to a long-term distribution of end-to-end delays, and characterizing their components to plan the appropriate action. Alternatively, the module could maintain a running predictor, continuously predicting future delays from a set of samples. This latter approach allows faster reaction to the variations that exist within the logical states, thus possibly achieving better performance compared to the former approach.

9.4.3 Control

The Control module transforms the features produced by the Analysis module into decisions. In this context, decisions control the activation/deactivation of pipelines, as well as probing strategies. For instance, a perceived degradation in the performance of the currently used pipeline, the Control module could activate probing over available pipelines. Minimizing

Δ, the capture-to-control delay requires continuous tuning of system parameters, such as sampling frequency, offloading strategy and analysis functions. Note that multiple pipelines could be kept active during transitions to avoid mission interruptions.

9.5 Application and Preliminary Discussion

We make our discussion specific to a relatively simple, but representative, case-study, where the task of the UAV is that of tracking objects in a predetermined class *e.g.*, pedestrians or vehicles. Figure 9.5 depicts the schematics of the application. The UAV acquires images at a constant rate. Each image is analyzed using an object detection algorithm to detect classes of objects of interest. The bounding box of a selected object detected by the algorithm is used to determine the motion of the UAV. Specifically, a control function computes the vector from the center of the image to the center of the corresponding bounding box. The vector is, then, converted into speed and direction commands.

We define the time from image capture to the emission of the control *capture-to-control delay*. Intuitively, minimizing the capture-to-output delay is critical to achieve a fast response of the UAV to movements of the target object and an effective tracking. The main bottleneck of the sensing-analysis-control pipeline is the analysis algorithm. State of the art object detection is realized via sophisticate algorithms, such as Deep Neural Networks (DNN). An example is RetinaNet [131], which features Pyramid Network backbone on top of a feedforward ResNet architecture (50 to 100 layers depending on the version). These DNNs provide a high accuracy both in terms of classification and precision of the bounding box, thus enabling accurate navigation in the considered application.

However, the execution of such complex algorithms on constrained embedded devices may be simply unfeasible, or result into excessive capture-to-control delay and energy consump-

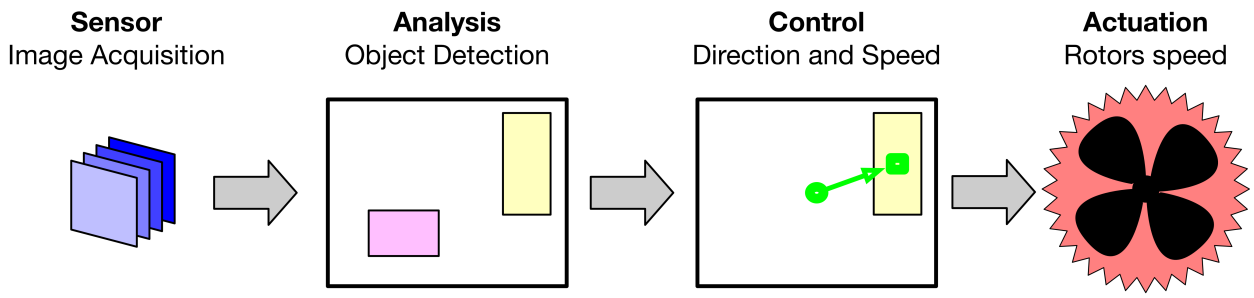


Figure 9.5: Application considered in the experiments: an autonomous UAV captures images from an onboard camera. An object detection algorithm is used to detect objects of interest. The bounding box is used to derive speed and direction and control the rotors.

tion. Simplified versions of these DNNs reduce computational complexity at the price of a degraded performance. Yolo [168] is an example of this kind of approach, a highly optimized software which degrades the accuracy of the output, especially in terms of bounding box precision. Simplified DNNs specifically tailored to mobile devices are also available [102, 104] corresponding to various points in the complexity-accuracy tradeoff.

Offloading these compute-intense tasks of the pipeline to interconnected compute-capable devices, such as edge servers, can significantly reduce the capture-to-control delay and decrease energy intake. However, offloading requires the wireless transfer of the signal/s produced by the onboard sensors and the output remotely produced. In the considered application scenario, the captured images from the onboard sensor are transferred at regular intervals to the server, which executes the DNN (analysis) and computes the control commands (control) that are then sent to the UAV.

As mentioned earlier, offloading introduces a possibly considerable degree of uncertainty:

- **Wireless link:** The continuous flow of images and control messages is transported over a wireless link, whose short-term capacity is influenced by time-varying factors such as channel gain, interference, network load, and fine-grain protocol interactions.
- **Server:** The time frame of execution of the task at the server is a function of time-

varying load, as well as of resource allocation strategies.

The objective of this study is that of analyzing the capture-to-control delay patterns over different communication technologies in a real-world platform.

9.6 Experimental Platform

The experimental platform we developed realizes the application described in the previous section over a distributed wirelessly interconnected system. In the following, we describe the hardware and software components.

9.6.1 Hydra and Hardware

We briefly describe the overall system our development effort is aiming at to provide some context to this study. HyDRA – Resilient computing for Heterogeneous Autonomous Devices – is a middleware architecture enabling the dynamic migration within interconnected systems of modules composing pipelines for autonomy. The objective is to maintain a capture-to-control delay as low as possible, while minimizing the amount of channel and computation resources used. The system is hierarchical and composed of three layers: UAV-users, UAV-servers and ground servers.

We aim to explore various processing and communication options. Therefore, we equip the UAVs and ground servers with an embedded device and a Software Defined Radio (SDR).

- **UAV-User:** A lightweight 3DR Solo UAV is equipped with a gimbal and a GoPro camera. A 3D printed structure is added to support an Up Board board, an Ettus Research USRP B200mini and a Lithium High Voltage (11.4V/5.2 Ah) battery. The

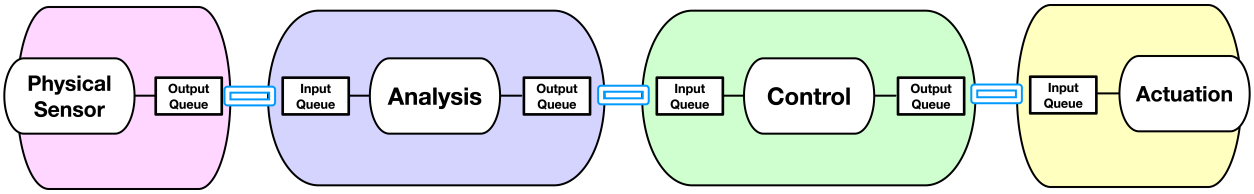


Figure 9.6: Modular architecture developed for the experiments. The modules wrap functions adding input/output functions for routing and filtering.

default on-board computer (IMX6) on the 3DR solo is only used to expose the Telemetry2 ports of the Pixhawk flight controller to be connected to UPboard via USB serial. The UP Board runs Ubuntu 16.04 OS to support all the on-board computing software and also the software suite for radio communications.

- **UAV-Server:** Large UAV International X6S *UAVs* are equipped with a Jetson NVIDIA TX2 and an Ettus Research USRP B210. The Jetson TX2 board has an NVIDIA Pascal-family GPU for high performance computation, and runs Ubuntu 16.04 with Kernel 4.4.

- **Ground Station - Server:** Ground edge servers are realized using a laptop with Intel Core i7-7700HQ CPU (8 cores, 2.8 GHz) and 16 GB RAM connected to the access point or base station.

All the devices are also equipped with a Wi-Fi dongle to enable Wi-Fi communication capabilities.

9.6.2 Software

Fig. 9.6 shows the modular pipeline developed to allow the distribution of analysis and control functions over the interconnected system. Each module encapsulates a function, and includes an input and an output queue. The modules logically abstract the functions composing the pipeline, isolating them from adjacent operations. Modules are atomic and can be

interchanged easily, and even dynamically. The modularization is necessary to achieve multi-threaded parallelization in-device. Naturally, modules that interact with the environment do not have "data" input (*e.g.* Sensor) or output (*e.g.*, Actuation).

The input and output queues provide flexibility to the flow of information throughout the pipeline. We remark that in our study some input and output queues and interconnected through sockets latched to the Wi-Fi or LTE stack. Queueing is critically necessary due to uncertainty in the time needed to traverse modules. For instance, as the capacity of the link connecting the UAV-user to a server changes, the timing of image forwarding may increase, so that queueing is necessary. Herein, we set the queue size to 1, and implement an aggressive preemptive policy at all modules, where new captured images, analysis output and control take the place of older one stored in the queue. This allows the minimization of the accumulation delay. Logging for the evaluation of performance metrics, such as the capture-to-control delay, is performed at the input and output queues. Specifically, the capture-to-control delay is the time interval delimited by the insertion of the data point into the Sensor's output queue and the insertion of the control message into the respective output queue. The complete HyDRA architecture includes dynamic routing within multiple pipelines and a logic controlling the routing paths based on system logs.

For LTE communications, the user UAVs are configured as User Equipment (UE) using srsLTE version 18.03 compliant with 3GPP release 8. We use srsLTE version 18.12 configured as eNodeB and core network (EPC) at the airborne and ground servers.

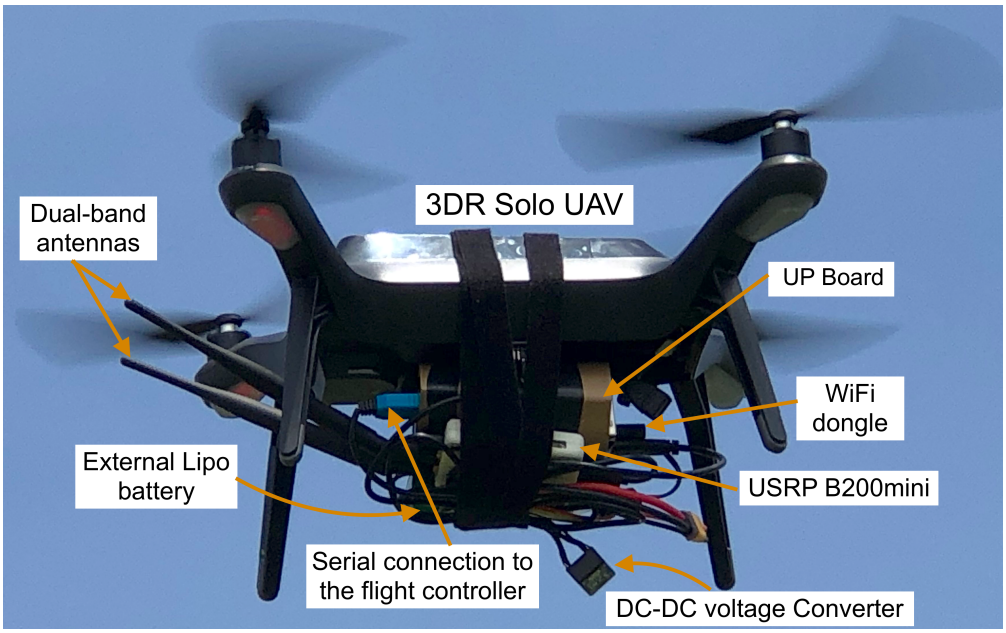


Figure 9.7: Hardware setup of 3DR solo UAV with all peripherals

9.7 Experimental Results

9.7.1 Setup

Figure 9.7 shows the setup of the UAV with the hardware components used for communication, computation and control. The experiments were conducted in a large outdoor environment. The UAV and ground server were connected using Wi-Fi (through Wi-Fi dongles) or LTE (through programmable radios implementing srsLTE).

The movements of the UAV are fully autonomous based on a pre-programmed mission on the flight controller. In order to characterize the application over capture-to-control delay over space and time, we disabled the function allowing the UAV to follow an object and, instead, programmed a predefined flight plan. Specifically, the UAV moves autonomously in steps to reach a specific distance and hovers for 1 minute at each distance.

We focus on a topology with a single UAV and ground server. The ground server runs

Parameters	Value
WiFi Standards	IEEE 802.11b and 802.11n
WiFi Bandwidth	20 MHz
WiFi Band	Channel 1 (2.412 GHz)
LTE Downlink EARFCN	3400
LTE Bandwidth	10 MHz
LTE Antenna Model	SISO
LTE UE Tx antenna gain	110 dB
LTE eNodeB Rx antenna gain	40 dB
LTE RLC Mode	Acknowledgement (RLC AM)
LTE MAC Scheduler	Round Robin
TCP Congestion Control	Cubic

Table 9.1: Experiment Parameters used for WiFi and LTE

on the same laptop that is configured as AP or base station, thus ensuring minimal delay between reception and processing. The communication parameters used for WiFi and LTE experiments are reported in the table 9.1.

In order to measure the impact of external traffic on the delay patterns, we set a second stationary user in the vicinity of the access point (AP) or the base station and generate periodic large data frames at different frequencies – corresponding to different data rates – over TCP. We remark that the area was not isolated from external communications and interference.

In the experiments, the sensor module acquires images (480×640 RGB pixels, compressed using JPEG to 26 KB). The analysis module executes a lightweight DNN for object detection (SSD-lite mobilenet v2 2018-09). Herein, we limit our study to pipelines originating from UAV-users and positioning analysis at the ground station-servers. The execution of the DNN model takes approximately 1 and 0.07 s on the Up Board and ground server, respectively.

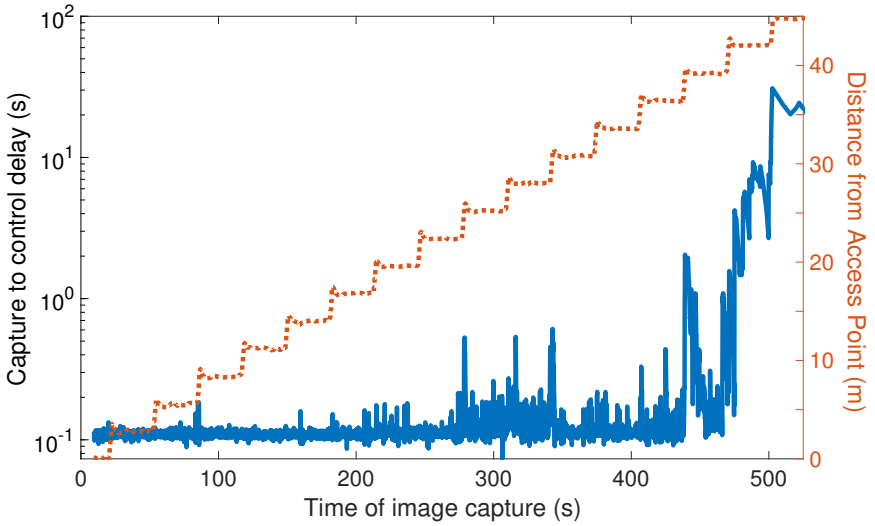


Figure 9.8: Capture-to-control delay over WiFi 802.11n. In this flight, the UAV-User moves away from the ground base station in free space.

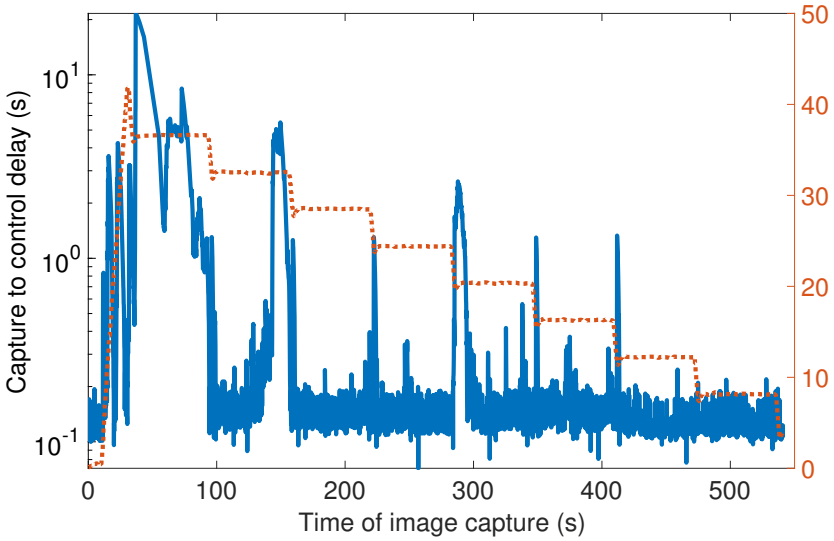


Figure 9.9: Capture-to-control delay over WiFi 802.11n. In this flight, the UAV-User moves toward the ground base station in free space. An external node produces traffic at 15 Mbps.

9.7.2 Results

Fig. 9.8 shows the temporal trace of the capture-to-control delay (blue line) using IEEE 802.11n obtained as the UAV flies away from the ground station. The orange line reports the distance from the ground station based on the received telemetry. It can be seen that when the distance is sufficiently small, the high throughput of the IEEE 802.11n grants a total

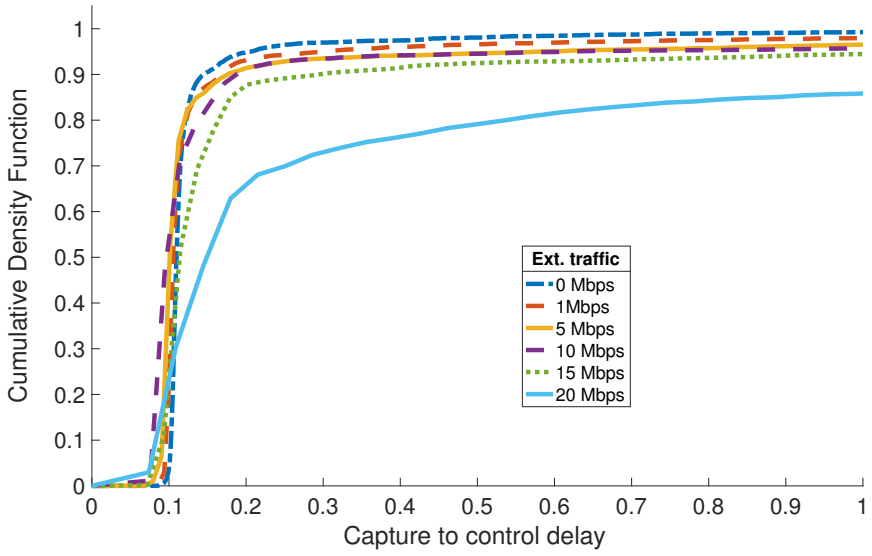


Figure 9.10: Cumulative density function of the capture-to-control delay for different volumes of external traffic.

delay of approximately 0.1 s, composed of approximately 0.07 s to execute object detection at the ground station and 0.03 s to transport the image and the control over wireless. As the distance increases both the average delay and its variance grow until disconnection occurs at about 35 m.

Fig. 9.15 shows the same trace – this time with the UAV approaching the ground station – when another node is active in the system producing traffic at 15 Mbps. We observe a generally increased delay floor (approximately 0.15 s average) and a much higher overall delay variance. High peaks of delay may correspond to TCP timeouts due to the interference load. Interestingly, sharp spikes are also present in correspondence with UAV acceleration from a stationary position. Similar, but less prominent, spikes could be noted in the previously shown trace, with a possibly faster "absorption" from the system due to the typically smaller delay. Although we do not have an explanation supported by conclusive evidence, the spikes could be generated by antenna misalignment when the UAV tilts to accelerate, or temporary large channel estimation error.

Fig. 9.10 shows the Cumulative Density Function (CDF) of the delay for different traffic

rates of the external node. The CDF is calculated by taking the average over each second, and then computing the CDF over the obtained delay. This to have a more fair weight in the CDF. As expected, a general degradation of the delay as contention increases is present. Interestingly, the contention scenarios in the $0 - 10$ Mbps range share a step-like shape of the CDF, where the key difference is the probability of high-delay spikes. Higher volumes of the contending data stream reduce the steepness of the step, introducing mid-range delays.

Fig. 9.11 compares the CDF achieved when using IEEE 801.11n and IEEE 801.11b. The lower maximum throughput offered by the latter is apparent. With no external traffic, the IEEE 801.11b suffers a small degradation in the minimum delay compared to IEEE 801.11n, due to the fact that the communication time is a small part of the total delay. However, 1 Mbps of external traffic are sufficient to dramatically increase the delay, shifting a substantial portion of delays to the $1 - 2$s range.

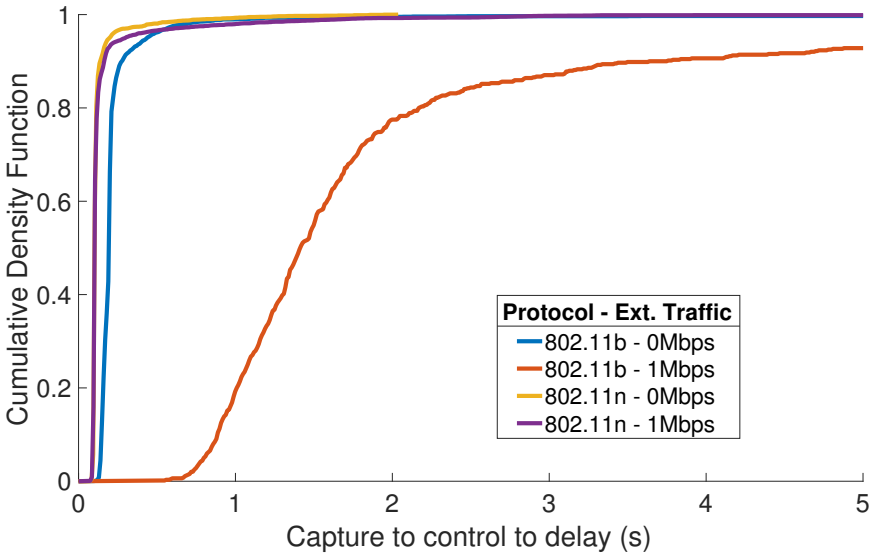


Figure 9.11: Cumulative density function of the capture-to-control delay for different volumes of external traffic and IEEE 801.11 versions .

We show in Fig. 9.12, which depics mean and variance of capture-to-control delay for IEEE 802.11n as a function of the external traffic, that not only the mean of the delay increases, but also its variance.

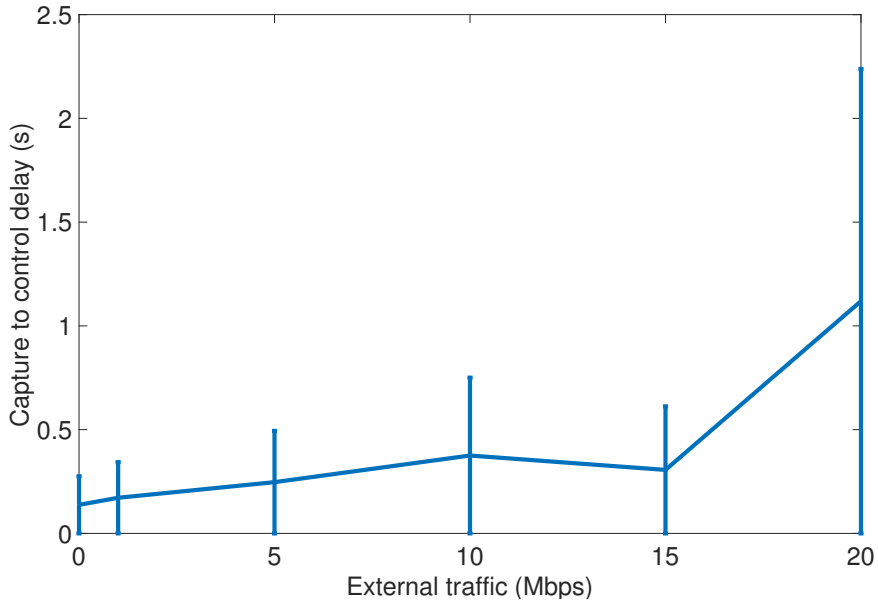


Figure 9.12: Average and standard deviation of capture to control delay, taken over a sequence of flying experiments, each with a constant external traffic rate. Distance from Access Point between 0-35 m.

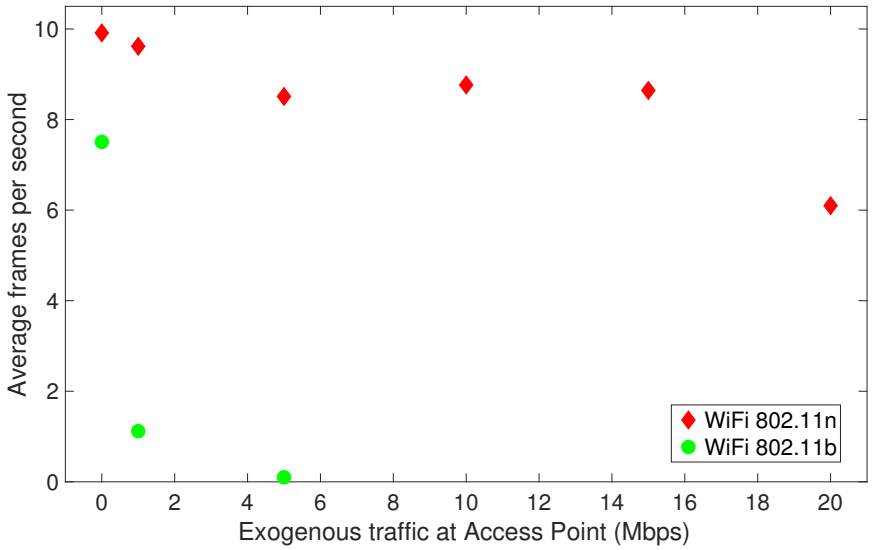


Figure 9.13: Average number of frames per second processed by the system for different IEEE 802.11 protocols and external traffic rates.

Fig. 9.13 shows the average number of frames per second processed by the system. The markers correspond to different traffic rates and IEEE 802.11 versions (n in red and b in green). We remark that while the emission rate of frames from the sensor is fixed, the preemptive queueing policy may eliminate frames from any queue, thus subsampling the

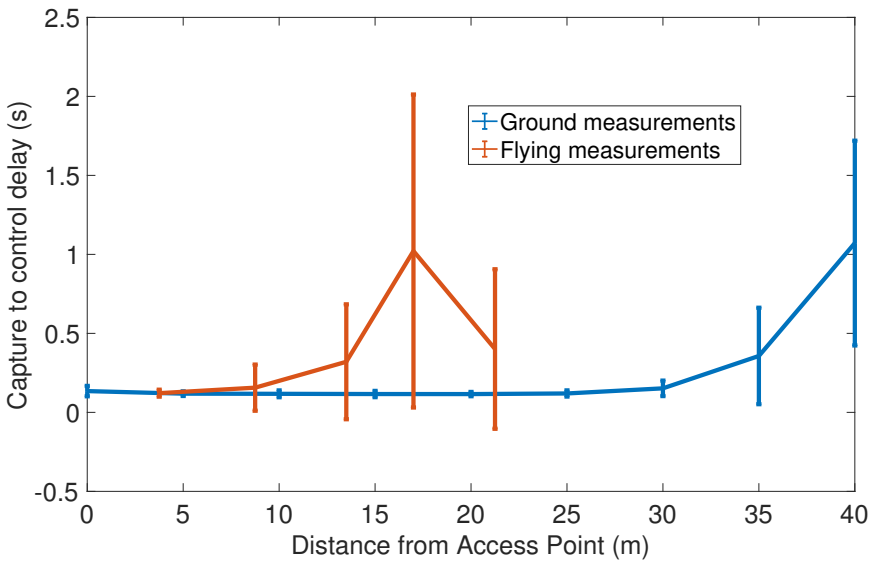


Figure 9.14: Mean and standard deviation of capture-to-control delay using Software Defined Radios emulating an LTE network. The blue and red lines correspond to stationary measurements on the ground and actual flight.

generated sequence as it is transformed. The high sensitivity of IEEE 802.11b to external traffic is apparent, making this protocol unsuitable to establish stable pipelines. Due to its high throughput, the IEEE 802.11n provides stable $8-10$ frames per second until degradation occurs at around 20 Mbps. We observe that the on-board processing pipeline would sustain about 1 frame per second.

Fig. 9.14 shows the mean and variance of delay achieved when using LTE over SDRs as a function of the distance. The blue and red lines correspond to stationary measurements on the ground and actual flight. Interestingly, it can be observed a substantial difference between ground measurements and flight. This may be caused by several factors including antenna emission patterns, UAV motion, and a larger interference from active LTE base stations perceived at a higher altitude. We observe that the SDRs have power constraints which degrade the system performance compared to actual cellular systems, this limits the range to tens of meters – vs hundreds of meters in commercial LTE systems. The trace reported in Fig. 9.15 shows a rapidly increasing number of delay spikes as the distance between the UAV and the ground server increases. We note that the delay floor is at about

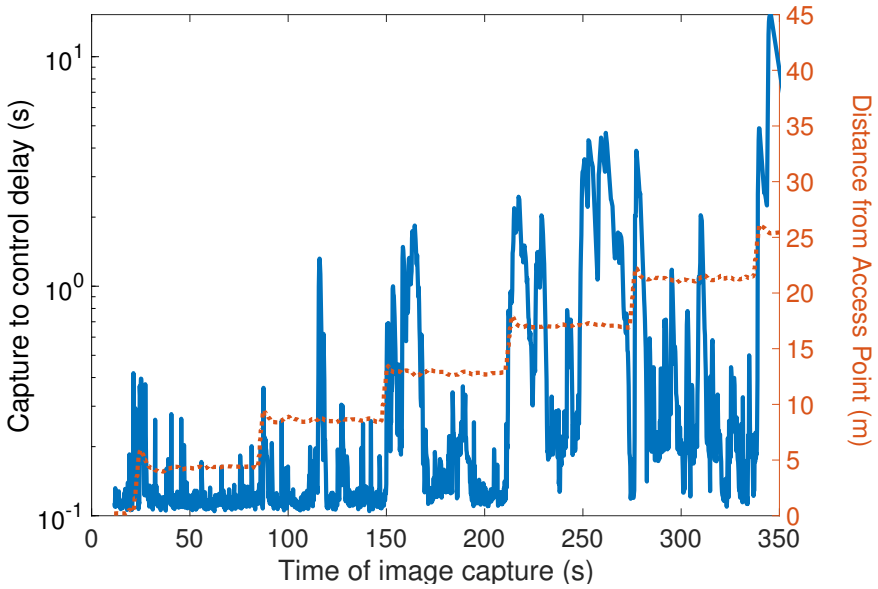


Figure 9.15: Capture to control time of the distributed pipeline, over LTE with Software Defined Radios.

0.15 s, and disconnection occurs at 20 m. We remark that experiments are not performed in an isolated environment, and some high-delay sections may be caused by other active LTE emitters. Future investigations will use other radios to measure the incoming energy in the used frequency band and correlate it with delay.

9.7.3 Discussion

The results showed in this section emphasized one critical aspect of offloading in this class of applications: wireless links from airborne devices are extremely unstable. Sensitivity to distance, physical orientation and exogeneous environmental conditions – such as other active data streams – was observed in the experiments. In Wi-Fi-based communications, interactions between Medium Access Control (MAC) and transport layer protocols of different devices increase delay and delay variance at the small and large scale of spatio-temporal trajectories. LTE offers a more stable multiplexing strategy. However, at least in the SDR-based setup used in our testbed, the delay patterns show a high variance, possibly motivated

by physical layer factors such as uncontrolled inter-cell interference, antenna alignment and limited power.

Clearly, relying on one communication-computation loop to perform critical tasks can greatly harm the overall ability to reliably control the UAV. This further motivates our effort to develop a system capable of moving sensing-processing-control pipelines dynamically across resources in response to perceived changes in the environment. Importantly, the step-like behavior of most delay's CDFs corresponds to clustered delay regions, and temporal traces make apparent that higher delays often appear in the form of short-term spikes. The general intuition, then, is that most coarse-grain environmental parameters are poor predictors of instantaneous delays, and that building strategies capable of overcoming the temporal variability of wireless channels in this class of systems is a relevant technical challenge.

9.8 Conclusions

Motivated by the development of Hydra, an architecture for resilient collaborative processing over hierarchical UAV systems, we presented a measurement study on the performance achieved migrating computing tasks from an autonomous UAV to an interconnected server. We specifically focused on capture-to-control delay in a class of applications focused on object detection-based navigation. Our setup includes various communication options: IEEE 801.11b and n and LTE emulated on SDRs using the srsLTE open-source software. The collected measurements emphasize the instability of the links during flight, and the need for a further layer of intelligence to dynamically select the best available option.

Chapter 10

Summary & Future Work

10.1 Summary of the Dissertation

In this dissertation we addressed the challenges in communication for intelligent and autonomous systems and provide different strategies for adaptation for fair coexistence and also maintaining the quality of service or computation. We specifically focused on mission-critical applications where real-time computation and/or control loops are involved. We presented different approaches to mitigate the issues of interference, bandwidth and latency constraints, scalable computations, predictive selection of communication channels. We experimented with real-world use cases, e.g. video streaming, object detection in real-time, UAV controls and communication, and applied the proposed solutions to improve the performance. Our results shows that the adaptive solutions should be incorporated or overlaid on the existing systems in order to support mission-critical and future applications.

10.2 Future Research Directions

We presented several adaptive communication approaches and demonstrated their benefits on real-world use case scenarios. As follow-up of our works, several interesting future research directions open up.

10.2.1 Collaborative Vision over V2X Communications

As we presented a case study for edge assisted object detection over LTE in coexisting scenario, the study can be extended for vehicle-to-everything (V2X) applications. For example, as a collaborative vision for transportation assistance using video stream from road-side camera or a video from drone. The assistance is especially beneficial when an object is occluded due to some objects and thus not noticeable by vehicles or pedestrians. The UAV can send the videos to the edge server over LTE which runs the object detection algorithms and notify the results to the transportation entities. Here an adaptive approach for coexistence similar to what we described in Chapter 3 can be incorporated. Future works may incorporate multi-sensor data fusion to accurately determine the objects, locations and impacts to make more intelligent decisions.

10.2.2 Robust Computation and Communication over UAVs using Flying Edge

Another work which this thesis directs to is using flying edge computation over UAVs. So far we used smaller UAVs with resource constraints and a static edge server helping the UAV for computation offloading and maintains the QoS. However, due to all the mentioned challenges of UAV, a solution with fixed edge may not be optimal and it necessitates the scope of using

a Powerful UAV as a flying edge. In flying edge supporting computation of a swarm of UAVs over wireless communication is an interesting area to explore. Another motivation for using flying edge is its capability of being a flying base station. A moderately big UAV can mount radio interfaces with a form of dongle or software-defined radios, e.g., USRP to act as a base station. The flying capability of this base station can control the relative mobility and the distances between itself and UAVs performing data acquisition. The better network quality gives more opportunity for offloading while maintaining capture-to-control delay for airborne systems as mentioned in Chapter 9.

10.2.3 Hybrid Communications in UAV Swarm using Infrastructure and Device-to-device Assistance

As we mentioned in this dissertation the challenges of communication for UAVs specially in urban environment, the future swarm applications will be more challenging. However, the swarm-based tasks can leverage the infrastructure assistance as mentioned in Chapter 8 or the nodes in the swarm can communicate over device-to-device (D2D) and achieves the required computation over multi-hop communication. Future studies can be performed on the optimality of using the infrastructure or the device-to-device communications. For using infrastructure intelligently with all available heterogeneous networks, e.g., WiFi, LTE, a learning based network selection can be implemented by enhancing the model presented in Chapter 6. Also, intelligent handover mechanism can be incorporated to opportunistically activating the network interfaces to minimize the energy consumption.

Bibliography

[1] 3D robotics, DroneKit-Python Documentation, 2015, python.dronekit.io, retrieved december 13, 2015.

[2] 3D robotics, "https://3dr.com/solo-drone/".

[3] 3GPP TS 36.213, E-UTRA physical layer procedures.

[4] Ardupilot sitl: "http://ardupilot.org/dev/docs/sitl-simulator-software-in-the-loop.html".

[5] Erle Robotics, "www.erlerobotics.com".

[6] Extending lte advanced to unlicensed spectrum. *Qualcomm Incorporated, White Paper, December 2013*.

[7] "flair: Framework libre air" available from https://devel.hds.utc.fr/software/flair.

[8] H-SIM flight simulator: "http://www.h-sim.com".

[9] NAVIO2 Autopilot, "https://emlid.com/navio/".

[10] NS-3 open source network simulator under GNU GPLv2 license. *https://www.nsnam.org/*.

[11] SURF: Speeded up Robust Features, author=Bay, Herbert and Tuytelaars, Tinne and Van Gool, Luc, journal=Computer vision–ECCV 2006, pages=404–417, year=2006, publisher=Springer.

[12] 3GPP TR 36.843 feasibility study on LTE device to device proximity services - radio aspects. 2014.

[13] 3GPP TS 23.303, proximity-based services (prose); stage 2 (release 12), v.12.0.0. February 2014.

[14] N. Abbas, S. Taleb, H. Hajj, and Z. Dawy. A learning-based approach for network selection in wlan/3g heterogeneous network. In *Communications and Information Technology (ICCIT), 2013 Third International Conference on*, pages 309–313. IEEE, 2013.

[15] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang. Oboe: auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58. ACM, 2018.

[16] A.-S. Ali and O. Simeone. Energy-Efficient Resource Allocation for Mobile Edge Computing-Based Augmented Reality Applications. *IEEE Wireless Communications Letters*, 2017.

[17] E. Almeida, A. Cavalcante, R. Paiva, F. Chaves, F. Abinader, R. Vieira, S. Choudhury, E. Tuomaala, and K. Doppler. Enabling LTE/WiFi coexistence by LTE blank subframe allocation. *in Proc. 2013 IEEE International Conference on Communications (ICC '13), Budapest, Hungary, pp.5083-5088*, 2013.

[18] A. Alvarez, R. Orea, S. Cabrero, X. G. Pañeda, R. García, and D. Melendi. Limitations of Network Emulation with Single-Machine and Distributed ns-3. In *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*, page 67. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010.

[19] M. Alzenad, A. El-Keyi, F. Lagum, and H. Yanikomeroglu. 3-d placement of an unmanned aerial vehicle base station (uav-bs) for energy-efficient maximal coverage. *IEEE Wireless Communications Letters*, 6(4):434–437, 2017.

[20] R. Amorim, H. Nguyen, P. Mogensen, I. Z. Kovács, J. Wigard, and T. B. Sørensen. Radio channel modeling for uav communication over cellular networks. *IEEE Wireless Communications Letters*, 6(4):514–517, 2017.

[21] G. Ananthanarayanan, P. Bahl, P. Bodík, K. Chintalapudi, M. Philipose, L. Ravindranath, and S. Sinha. Real-time video analytics: The killer app for edge computing. *computer*, 50(10):58–67, 2017.

[22] S. Andreev, M. Gerasimenko, O. Galinina, Y. Koucheryavy, N. Himayat, S.-P. Yeh, and S. Talwar. Intelligent access network selection in converged multi-radio heterogeneous networks. *IEEE Wireless Communications*, 21(6):86–96, 2014.

[23] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro. Lte for vehicular networking: a survey. *IEEE communications magazine*, 51(5):148–157, 2013.

[24] ArduPilot. SITL Simulator (Software in the Loop). 2016.

[25] H. Arslan. *Cognitive Radio, Software Defined Radio, and Adaptive Wireless Systems*, volume 10. Springer, 2007.

[26] A. Asadi and V. Mancuso. Wifi direct and lte d2d in action. In *2013 IFIP Wireless Days (WD)*, pages 1–8. IEEE, 2013.

[27] A. Asadi, Q. Wang, and V. Mancuso. A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys & Tutorials*, 16(4):1801–1819, 2014.

[28] D. Athukoralage, I. Guvenc, W. Saad, and M. Bennis. Regret based learning for uav assisted lte-u/wifi public safety networks. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2016.

[29] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[30] A. Babaei, J. Andreoli-Fang, Y. Pang, and B. Hamzeh. On the impact of lte-u on wi-fi performance. *International Journal of Wireless Information Networks*, 22(4):336–344, 2015.

[31] L. Badia, N. Baldo, M. Levorato, and M. Zorzi. A markov framework for error control techniques based on selective retransmission in video transmission over wireless channels. *IEEE Journal on Selected Areas in Communications*, 28(3):488–500, 2010.

[32] L. Badia, M. Levorato, and M. Zorzi. Analysis of selective retransmission techniques for differentially encoded data. *IEEE International Conference on Communications - ICC*, 2009.

[33] S. Baidya and M. Levorato. Edge-assisted Computation-Driven Dynamic Network Selection for Real-Time Services in the Urban IoT. In *IEEE International Workshop on Advances in Software Defined and Context Aware Cognitive Radio Networks (IEEE SCAN-2017)*.

[34] S. Baidya and M. Levorato. Content-based cognitive interference control for city monitoring applications in the urban iot. In *Global Communications Conference (GLOBECOM), 2016 IEEE*, pages 1–6. IEEE, 2016.

[35] S. Baidya and M. Levorato. Content-based Cognitive Interference Control for City Monitoring Applications in the Urban IoT. *IEEE Global Communications Conference, GLOBECOM 2016, Washington DC, USA, December 4-8*, 2016.

[36] S. Baidya and M. Levorato. Content-based Cognitive Interference Control for City Monitoring Applications in the Urban IoT. *IEEE Global Communications Conference, GLOBECOM 2016, Washington DC, USA, December 4-8*, 2016.

[37] S. Baidya and M. Levorato. Content-based Interference Management for Video Transmission in D2D Communications underlaying LTE. In *International Conference on Computing, Networking and Communications (ICNC), 2017*, pages 144–149. IEEE, 2017.

[38] S. Baidya, Z. Shaikh, and M. Levorato. FlyNetSim: An Open Source Synchronized UAV Network Simulator based on ns-3 and Ardupilot. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 37–45. ACM, 2018.

[39] S. Baidya, Z. Shaikh, and M. Levorato. Flynetsim: Flying and networking simulator. `https://github.com/saburhb/FlyNetSim`, 2018.

[40] 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.

[41] S. Barré, C. Paasch, and O. Bonaventure. Multipath tcp: from theory to practice. In *International Conference on Research in Networking*, pages 444–457. Springer, 2011.

[42] F. Bashir and F. Porikli. Performance evaluation of object detection and tracking systems. *Proceedings 9th IEEE International Workshop on PETS. 2006*, 2006.

[43] H. Baya, A. Essa, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *Computer Vision and Image Understanding, Volume 110, Issue 3, June 2008, Pages 346-359*, June 2008.

[44] A. Begel, S. McCanne, and S. L. Graham. Bpf+: Exploiting Global Data-Flow Optimization in a Generalized Packet Filter Architecture. In *ACM SIGCOMM Computer Communication Review*, volume 29, pages 123–134. ACM, 1999.

[45] I. Bekmezci, O. K. Sahingoz, and Ş. Temel. Flying ad-hoc networks (fanets): A survey. *Ad Hoc Networks*, 11(3):1254–1270, 2013.

[46] F. Bellard, M. Niedermayer, et al. Ffmpeg. *Availabel from: http://ffmpeg. org*, 2012.

[47] P. Bellavista, G. Cardone, A. Corradi, and L. Foschini. Convergence of MANET and WSN in IoT urban scenarios. *IEEE Sensors Journal*, 13(10):3558–3567, 2013.

[48] D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 2. Athena Scientific, Belmont, MA, 2nd edition, 2001.

[49] A. R. Biswas and R. Giaffreda. Iot and cloud convergence: Opportunities and challenges. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 375–376. IEEE, 2014.

[50] B.Kaufman and B.Aazhang. Cellular networks with an overlaid device to device network. *in Proc. Asilomar Conf. Signals, Syst. Comput., 2008, pp. 1537-1541*, 2008.

[51] F. Boccardi, R. Heath, A. Lozano, T. Marzetta, and P. Popovski. Five disruptive technology directions for 5g. *IEEE Communications Magazine*, 52(2):74–80, 2014.

[52] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.

[53] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, 2012.

[54] A. Botta, W. De Donato, V. Persico, and A. Pescapé. On the integration of cloud computing and internet of things. In *International Conference on Future Internet of Things and Cloud (FiCloud), 2014*, pages 23–30. IEEE, 2014.

[55] M. Bramberger, J. Brunner, B. Rinner, and H. Schwabach. Real-time video analysis on an embedded smart camera for traffic surveillance. In *10th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 174–181. IEEE, 2004.

[56] H. Cai, B. Xu, L. Jiang, and A. V. Vasilakos. Iot-based big data storage systems in cloud computing: perspectives and challenges. *IEEE Internet of Things Journal*, 4(1):75–87, 2016.

[57] D. Callegaro and M. Levorato. Optimal computation offloading in edge-assisted uav systems. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[58] D. Callegaro and M. Levorato. Optimal computation offloading in Edge-Assisted UAV systems. In *2018 IEEE Global Communications Conference: Selected Areas in Communications: Tactile Internet (Globecom2018 SAC TI)*, Abu Dhabi, United Arab Emirates, Dec. 2018.

[59] N. Cao, S. B. Nasir, S. Sen, and A. Raychowdhury. Self-optimizing iot wireless video sensor node with in-situ data analytics and context-driven energy-aware real-time adaptation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2470–2480, 2017.

[60] X. Cao, J. Xu, and R. Zhangt. Mobile edge computing for cellular-connected uav: Computation offloading and trajectory optimization. In *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2018.

[61] X. Chang. Network Simulations with OPNET. In *Proceedings of the 31st conference on Winter simulation: Simulation—a bridge to the future-Volume 1*, pages 307–314. ACM, 1999.

[62] N. Chen, Y. Chen, Y. You, H. Ling, P. Liang, and R. Zimmermann. Dynamic urban surveillance video stream processing using fog computing. In *2016 IEEE second international conference on multimedia big data (BigMM)*, pages 105–112. IEEE, 2016.

[63] Q. Chen, F. R. Yu, T. Huang, R. Xie, J. Liu, and Y. Liu. Integrated Resource Management in Software Defined Networking, Caching and Computing. *arXiv preprint arXiv:1611.05122*, 2016.

[64] X. Chen, L. Jiao, W. Li, and X. Fu. Efficient multi-user computation offloading for mobile-edge cloud computing. *IEEE/ACM Transactions on Networking*, (5):2795–2808, 2016.

[65] P. Cheng, L. Deng, H. Yu, Y. Xu, and H. Wang. Resource allocation for cognitive networks with d2d communication: An evolutionary approach. In *Wireless Communications and Networking Conference (WCNC), 2012 IEEE*, pages 2671–2676. IEEE, 2012.

[66] C.-M. Chou, C.-Y. Li, W.-M. Chien, and K.-c. Lan. A feasibility study on vehicle-to-infrastructure communication: Wifi vs. wimax. In *2009 tenth international conference on mobile data management: systems, services and middleware*, pages 397–398. IEEE, 2009.

[67] M. Condoluci, L. Militano, A. Orsino, J. Alonso-Zarate, and G. Araniti. Lte-direct vs. wifi-direct for machine-type communications over lte-a systems. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2298–2302. IEEE, 2015.

[68] G. Convertino, D. Melpignano, E. Piccinelli, F. Rovati, and F. Sigona. Wireless adaptive video streaming by real-time channel estimation and video transcoding. In *International Conference on Consumer Electronics, 2005*, pages 179–180. IEEE, 2005.

[69] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 181–192. ACM, 2012.

[70] J. Dick, C. Phillips, S. H. Mortazavi, and E. de Lara. High speed object tracking using edge computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 26. ACM, 2017.

[71] K. Dolui and S. K. Datta. Comparison of edge computing implementations: Fog computing, cloudlet and mobile edge computing. In *2017 Global Internet of Things Summit (GIoTS)*, pages 1–6. IEEE, 2017.

[72] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl. Device-to-device communication as an underlay to lte-advanced networks. *IEEE Communications Magazine*, 47(12):42–49, 2009.

[73] K. Doppler, M. Rinne, C. Wijting, C. B. Ribeiro, and K. Hugl. Device-to-device communication as an underlay to lte-advanced networks. *IEEE Communications Magazine*, 47(12):42–49, Dec 2009.

[74] K. Doppler, C.-H. Yu, C. B. Ribeiro, and P. Janis. Mode selection for device-to-device communication underlaying an lte-advanced network. In *Wireless Communications and Networking Conference (WCNC), 2010 IEEE*, pages 1–6. IEEE, 2010.

[75] C. Doukas and I. Maglogiannis. Bringing iot and cloud computing towards pervasive healthcare. In *2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pages 922–926. IEEE, 2012.

[76] M. Erdelj and E. Natalizio. Uav-assisted disaster management: Applications and open issues. In *2016 international conference on computing, networking and communications (ICNC)*, pages 1–5. IEEE, 2016.

[77] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. Help from the sky: Leveraging uavs for disaster management. *IEEE Pervasive Computing*, 16(1):24–32, 2017.

[78] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM computing surveys (CSUR)*, 35(2):114–131, 2003.

[79] P. Ferrari, A. Flammini, M. Loda, S. Rinaldi, D. Pagnoncelli, and E. Ragaini. First experimental characterization of lte for automation of smart grid. In *2015 IEEE International Workshop on Applied Measurements for Power Systems (AMPS)*, pages 108–113. IEEE, 2015.

[80] A. Filieri, M. Maggio, K. Angelopoulos, N. D'Ippolito, I. Gerostathopoulos, A. B. Hempel, H. Hoffmann, P. Jamshidi, E. Kalyvianaki, C. Klein, F. Krikava, S. Misailovic, A. V. Papadopoulos, S. Ray, A. M. Sharifloo, S. Shevtsov, M. Ujma, and T. Vogel. Software engineering meets control theory. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '15, pages 71–82, Piscataway, NJ, USA, 2015. IEEE Press.

[81] F. Fitzek and M. Reisslein. Mpeg-4 and h. 263 video traces for network performance evaluation. *IEEE Network*, 15(6):40–54, 2001.

[82] O. Flauzac, C. González, A. Hachani, and F. Nolot. Sdn based architecture for iot and improvement of the security. In *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, pages 688–693. IEEE, 2015.

[83] G. Fodor, E. Dahlman, G. Mildh, S. Parkvall, N. Reider, G. Miklós, and Z. Turányi. Design aspects of network assisted device-to-device communications. *IEEE Communications Magazine*, 50(3), 2012.

[84] G. Fodor and N. Reider. A distributed power control scheme for cellular network assisted d2d communications. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pages 1–6. IEEE, 2011.

[85] L. Foundation. Io visor project. *URL http://iovisor.org*, 2015.

[86] B. Furht and S. A. Ahson. *Long Term Evolution: 3GPP LTE radio and cellular technology*. Crc Press, 2016.

[87] P. Garcia Lopez, A. Montresor, D. Epema, A. Datta, T. Higashino, A. Iamnitchi, M. Barcellos, P. Felber, and E. Riviere. Edge-centric computing: Vision and challenges. *SIGCOMM Comput. Commun. Rev.*, 45(5):37–42, Sept. 2015.

[88] L. Garcia Paucar and N. Bencomo. Runtime models based on dynamic decision networks: enhancing the decision-making in the domain of ambient assisted living applications. In S. Götz, N. Bencomo, K. Bellman, and G. Blair, editors, *MRT 2016 - Models@run.time*, CEUR workshop proceedings, pages 9–17. CEUR-WS.org, 11 2016.

[89] E. Gaura, J. Brusey, M. Allen, R. Wilkins, D. Goldsmith, and R. Rednic. Edge mining the internet of things. *Sensors Journal, IEEE*, 13(10):3816–3825, Oct 2013.

[90] S. Geirhofer, L. Tong, and B. M. Sadler. Cognitive radios for dynamic spectrum access-dynamic spectrum access in the time domain: Modeling and exploiting white space. *IEEE Communications Magazine*, 45(5), 2007.

[91] M. Gerasimenko, N. Himayat, S.-p. Yeh, S. Talwar, S. Andreev, and Y. Koucheryavy. Characterizing performance of load-aware network selection in multi-radio (wifi/lte) heterogeneous networks. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 397–402. IEEE, 2013.

[92] S. Greco, J. Figueira, and M. Ehrgott. *Multiple criteria decision analysis*. Springer, 2016.

[93] B. Gregg. ebpf: One small step. *URL http://www.brendangregg.com/blog/2015-05-15/ebpf-one-small-step.html*, May, 2015.

[94] Y. Gu, M. Zhou, S. Fu, and Y. Wan. Airborne wifi networks through directional antennae: An experimental study. In *2015 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1314–1319. IEEE, 2015.

[95] L. Gupta, R. Jain, and G. Vaszkun. Survey of important issues in uav communication networks. *IEEE Communications Surveys & Tutorials*, 18(2):1123–1152, 2015.

[96] B. Han, F. Qian, L. Ji, and V. Gopalakrishnan. Mp-dash: Adaptive video streaming over preference-aware multipath. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, pages 129–143. ACM, 2016.

[97] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal. Nfv: State of the Art, Challenges, and Implementation in Next Generation Mobile Networks (vEPC). *IEEE Network*, 28(6):18–26, 2014.

[98] S. Hayat, E. Yanmaz, and C. Bettstetter. Experimental analysis of multipoint-to-point uav communications with ieee 802.11 n and 802.11 ac. In *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 1991–1996. IEEE, 2015.

[99] W. He, G. Yan, and L. Da Xu. Developing vehicular data cloud services in the iot environment. *IEEE Transactions on Industrial Informatics*, 10(2):1587–1595, 2014.

[100] 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.

[101] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *AIAA guidance, navigation and control conference and exhibit*, page 6461, 2007.

[102] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[103] Y. C. Hu, M. Patel, D. Sabella, N. Sprecher, and V. Young. Mobile edge computinga key technology towards 5g. *ETSI white paper*, 11(11):1–16, 2015.

[104] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017.

[105] R. Huang, J. Pedoeem, and C. Chen. Yolo-lite: A real-time object detection algorithm optimized for non-gpu computers. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 2503–2510. IEEE, 2018.

[106] Q. Huynh-Thu and M. Ghanbari. Scope of Validity of PSNR in Image/Video Quality Assessment. *Electronics letters*, 44(13):800–801, 2008.

[107] R. ITU-R. P. 1411-8,. *Propagation data and prediction methods for the planning of short-range outdoor radiocommunication systems and radio local area networks in the frequency range 300MHz to 100GHz*, page 11, 2015.

[108] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using sctp multihoming over independent end-to-end paths. *IEEE/ACM Transactions on networking*, 14(5):951–964, 2006.

[109] J. Mitola. Cognitive radio: an integrated agent architecture for software-defined radio. Doctor of Technology, Royal Inst. Technol. (KTH), Stockholm, Sweden, 2000.

[110] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne. Rtp: A transport protocol for real-time applications. 2003.

[111] P. Jänis, C.-H. Yu, K. Doppler, C. Ribeiro, C. Wijting, K. Hugl, O. Tirkkonen, and V. Koivunen. Device-to-device communication underlaying cellular communications systems. *Int. Journal of Communications, Network and System Sciences*, 2(3):169, 2009.

[112] I. Jawhar, N. Mohamed, J. Al-Jaroodi, D. P. Agrawal, and S. Zhang. Communication and networking of uav-based systems: Classification and associated architectures. *Journal of Network and Computer Applications*, 84:93–108, 2017.

[113] J. Jeon, H. Niu, Q. Li, A. Papathanassiou, and G. Wu. Lte with listen-before-talk in unlicensed spectrum. *In 2015 IEEE International Conference on Communication Workshop (ICCW), pp. 2320-2324*, 2015.

[114] J. Jin, J. Gubbi, T. Luo, and M. Palaniswami. Network Architecture and QoS Issues in the Internet of Things for a Smart City. In *International Symposium on Communications and Information Technologies (ISCIT), 2012*, pages 956–961. IEEE, 2012.

[115] T. Jordan, J. Foster, R. Bailey, and C. Belcastro. Airstar: A uav platform for flight dynamics and control system testing. In *25th AIAA Aerodynamic Measurement Technology and Ground Testing Conference*, page 3307, 2006.

[116] K. W. Ross. Randomized and past-dependent policies for Markov decision processes with multiple constraints. Operations Research, 37(3):474–477, May-June 1989.

[117] S. Kajita, H. Yamaguchi, T. Higashino, H. Urayama, M. Yamada, and M. Takai. Throughput and delay estimator for 2.4 ghz wifi aps: A machine learning-based approach. In *IFIP Wireless and Mobile Networking Conference (WMNC), 2015 8th*, pages 223–226. IEEE, 2015.

[118] N. Ketkar. Introduction to pytorch. In *Deep learning with python*, pages 195–208. Springer, 2017.

[119] J. Kim and S. Sukkarieh. Autonomous airborne navigation in unknown terrain environments. *IEEE Transactions on Aerospace and Electronic Systems*, 40(3):1031–1045, 2004.

[120] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[121] N. Koenig and A. Howard. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2004.(IROS 2004)*, volume 3, pages 2149–2154. IEEE, 2004.

[122] J. Kua, G. Armitage, and P. Branch. A survey of rate adaptation techniques for dynamic adaptive streaming over http. *IEEE Communications Surveys & Tutorials*, 19(3):1842–1866, 2017.

[123] S. G. Kulkarni, W. Zhang, J. Hwang, S. Rajagopalan, K. Ramakrishnan, T. Wood, M. Arumaithurai, and X. Fu. Nfvnice: Dynamic backpressure and scheduling for nfv service chains. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 71–84. ACM, 2017.

[124] R. Kumar, A. Francini, S. Panwar, and S. Sharma. Dynamic control of rlc buffer size for latency minimization in mobile ran. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6. IEEE, 2018.

[125] M. Lavassani, S. Forsström, U. Jennehag, and T. Zhang. Combining fog computing with sensor mote machine learning for industrial iot. *Sensors*, 18(5):1532, 2018.

[126] Y. LeCun, Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[127] P. Levis, N. Lee, M. Welsh, and D. Culler. Tossim: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 126–137. ACM, 2003.

[128] M. Levorato, D. O'Neill, A. Goldsmith, and U. Mitra. Optimization of ARQ protocols in interference networks with QoS constraints. In *IEEE International Conference on Communications (ICC)*, pages 1–6, 2011.

[129] H. Li, K. Ota, and M. Dong. Learning iot in edge: Deep learning for the internet of things with edge computing. *IEEE Network*, 32(1):96–101, 2018.

[130] J. C. Li, M. Lei, and F. Gao. Device-to-device (d2d) communication in mu-mimo cellular networks. In *Global Communications Conference (GLOBECOM), 2012 IEEE*, pages 3583–3587. IEEE, 2012.

[131] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[132] Y. Lin and Y. Hsu. Multihop cellular: A new architecture for wireless communications. *in Proc. IEEE INFOCOM, 2000, vol. 3, pp. 1273-1282*, 2000.

[133] J. Liu, N. Kato, J. Ma, and N. Kadowaki. Device-to-device communication in lte-advanced networks: A survey. *IEEE Communications Surveys & Tutorials*, 17(4):1923–1940, 2015.

[134] Z. Liu, T. Peng, S. Xiang, and W. Wang. Mode selection for device-to-device (d2d) communication under lte-advanced networks. In *IEEE International Conference on Communications (ICC), 2012*, pages 5563–5567. IEEE, 2012.

[135] A. Lombardo, A. Manzalini, G. Schembra, G. Faraci, C. Rametta, and V. Riccobene. An open framework to enable netfate (network functions at the edge). In *1st IEEE Conference on Network Softwarization (NetSoft), 2015*, pages 1–6. IEEE, 2015.

[136] E. Lte. Evolved universal terrestrial radio access (e-utra); base station (bs) radio transmission and reception (3gpp ts 36.104 version 8.6. 0 release 8), july 2009. *ETSI TS*, 136(104):V8, 2009.

[137] T. Lunttila, J. Lindholm, K. Pajukoski, E. Tiirola, and A. Toskala. Eutran uplink performance. In *2nd International Symposium on Wireless Pervasive Computing, 2007. ISWPC'07.* IEEE, 2007.

[138] C. Luo, J. Nightingale, E. Asemota, and C. Grecos. A uav-cloud system for disaster sensing applications. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5. IEEE, 2015.

[139] R. C. Luo, M.-H. Lin, and R. S. Scherp. Dynamic multi-sensor data fusion system for intelligent robots. *IEEE Journal on Robotics and Automation*, 4(4):386–396, 1988.

[140] J. Lyu, Y. Zeng, R. Zhang, and T. J. Lim. Placement optimization of uav-mounted mobile base stations. *IEEE Communications Letters*, 21(3):604–607, 2016.

[141] P. Mach and Z. Becvar. Mobile edge computing: A survey on architecture and computation offloading. *arXiv preprint arXiv:1702.05309*, 2017.

[142] J. Malinen et al. hostapd: Ieee 802.11 ap, ieee 802.1 x. Technical report, WPA/WPA2/EAP/RADIUS Authenticator. online: http://hostap. epitest. fi/hostapd, 2014.

[143] B. Manoj, R. R. Rao, and M. Zorzi. Cognet: a cognitive complete knowledge network system. *IEEE Wireless Communications*, 15(6):81–88, 2008.

[144] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210. ACM, 2017.

[145] Y. Mao, J. Zhang, S. Song, and K. B. Letaief. Stochastic Joint Radio and Computational Resource Management for Multi-User Mobile-Edge Computing Systems. *arXiv preprint arXiv:1702.00892*, 2017.

[146] E. A. Marconato, M. Rodrigues, R. d. M. Pires, D. F. Pigatto, C. Q. Luiz Filho, A. R. Pinto, and K. R. Branco. AVENS-A Novel Flying Ad Hoc Network Simulator with Automatic Code Generation for Unmanned Aircraft System. In *Proceedings of the 50th Hawaii International Conference on System Sciences*, 2017.

[147] S. McCanne and V. Jacobson. The BSD Packet Filter: A New Architecture for User-level Packet Capture. In *USENIX winter*, volume 46, 1993.

[148] L. Meier, J. Camacho, B. Godbolt, J. Goppert, L. Heng, M. Lizarraga, et al. Mavlink: Micro Air Vehicle Communication Protocol. *Online]. Tillgänglig: http://qgroundcontrol. org/mavlink/start.[Hämtad 2014-05-22]*, 2013.

[149] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. PIXHAWK: A Micro Aerial Vehicle Design for Autonomous Flight using Onboard Computer Vision. *Autonomous Robots, Springer*, 33(1-2):21–39, 2012.

[150] H. Menouar, I. Guvenc, K. Akkaya, A. S. Uluagac, A. Kadri, and A. Tuncer. Uav-enabled intelligent transportation systems for the smart city: Applications and challenges. *IEEE Communications Magazine*, 55(3):22–28, 2017.

[151] H. B. Mitchell. *Multi-sensor data fusion: an introduction*. Springer Science & Business Media, 2007.

[152] J. Mogul, R. Rashid, and M. Accetta. *The Packer Filter: An Efficient Mechanism for User-Level Network Code*, volume 21. ACM, 1987.

[153] R. Morabito, V. Cozzolino, A. Y. Ding, N. Beijar, and J. Ott. Consolidate iot edge computing with lightweight virtualization. *IEEE Network*, 32(1):102–111, 2018.

[154] H. Morkner, M. Karakucuk, G. Carr, and S. Espino. A full duplex front end module for wifi 802.11. n applications. In *2008 European Conference on Wireless Technology*, pages 162–165. IEEE, 2008.

[155] N. C. Narendra, K. Koorapati, and V. Ujja. Towards cloud-based decentralized storage for internet of things data. In *2015 IEEE International Conference on Cloud Computing in Emerging Markets (CCEM)*, pages 160–168. IEEE, 2015.

[156] E. Natalizio, D. Cavalcanti, K. Chowdhury, and M. El Said. Advances in Wireless Communication and Networking for Cooperating Autonomous Systems, Elsevier 2018.

[157] H. C. Nguyen, R. Amorim, J. Wigard, I. Z. Kovács, T. B. Sørensen, and P. E. Mogensen. How to ensure reliable connectivity for aerial vehicles over cellular networks. *Ieee Access*, 6:12304–12317, 2018.

[158] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab. Demo: Openairinterface: an open lte network in a pc. In *Proceedings of the 20th annual international conference on Mobile computing and networking*, pages 305–308. ACM, 2014.

[159] 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.

[160] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. *IEEE Communications Surveys & Tutorials*, 16(3):1617–1634, 2014.

[161] M. Ojo, D. Adami, and S. Giordano. A sdn-iot architecture with nfv implementation. In *2016 IEEE Globecom Workshops (GC Wkshps)*, pages 1–6. IEEE, 2016.

[162] R. Pantos and W. May. Http live streaming. Technical report, 2017.

[163] T. Peng, Q. Lu, H. Wang, S. Xu, and W. Wang. Interference avoidance mechanisms in the hybrid cellular and device-to-device systems. *in Proc. IEEE PIMRC, 2009, pp. 617-621*, 2008.

[164] B. Pfaff, J. Pettit, T. Koponen, E. Jackson, A. Zhou, J. Rajahalme, J. Gross, A. Wang, J. Stringer, P. Shelar, et al. The Design and Implementation of Open Vswitch. In *12th USENIX symposium on networked systems design and implementation (NSDI 15)*, pages 117–130, 2015.

[165] P. Phunchongharn, E. Hossain, and D. Kim. Resource allocation for device-to-device communications underlaying LTE-advanced networks. *IEEE Wireless Communications*, 20(4):91–100, 2013.

[166] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: An Open-Source Robot Operating System. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[167] B. Rao, H. F. Durrant-Whyte, and J. Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *The International Journal of Robotics Research*, 12(1):20–44, 1993.

[168] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[169] G. F. Riley and T. R. Henderson. The ns-3 Network Simulator. In *Modeling and tools for network simulation*, pages 15–34. Springer, 2010.

[170] H. Ritchie and M. Roser. Urbanization. *Our World in Data*, 2019. https://ourworldindata.org/urbanization.

[171] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[172] H. Schwarz, D. Marpe, and T. Wiegand. Overview of the scalable video coding extension of the h. 264/avc standard. *IEEE Transactions on circuits and systems for video technology*, 17(9):1103–1120, 2007.

[173] H. Seo, K.-D. Lee, S. Yasukawa, Y. Peng, and P. Sartori. Lte evolution for vehicle-to-everything services. *IEEE communications magazine*, 54(6):22–28, 2016.

[174] Z. Shaikh, S. Baidya, and M. Levorato. Robust multi-path communications for uavs in the urban iot. In *2018 IEEE International Conference on Sensing, Communication and Networking (SECON Workshops)*, pages 1–5. IEEE, 2018.

[175] S. Shevtsov, M. Berekmeri, D. Weyns, and M. Maggio. Control-theoretical software adaptation: A systematic literature review. *IEEE Transactions on Software Engineering*, 44(8):784–810, Aug 2018.

[176] 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.

[177] W. Shi and S. Dustdar. The promise of edge computing. *Computer*, 49(5):78–81, 2016.

[178] M. Soliman, T. Abiodun, T. Hamouda, J. Zhou, and C.-H. Lung. Smart home: Integrating internet of things with web services and cloud computing. In *2013 IEEE 5th international conference on cloud computing technology and science*, volume 2, pages 317–320. IEEE, 2013.

[179] G. Sona, D. Passoni, L. Pinto, D. Pagliari, D. Masseroni, B. Ortuani, and A. Facchi. UAV Multispectral Survey to map soil and crop for Precision Farming Applications. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 41:1023, 2016.

[180] A. Stanciu. Blockchain based distributed control system for edge computing. In *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pages 667–671. IEEE, 2017.

[181] R. R. Stewart and Q. Xie. Stream control transmission protocol (sctp). 2001.

[182] T. Stockhammer. Dynamic adaptive streaming over http–: standards and design principles. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 133–144. ACM, 2011.

[183] K. Stuhlmüller, N. Färber, M. Link, and B. Girod. Analysis of video transmission over lossy channels. *IEEE Journal on Selected Areas in Communications*, 18(6):1012–1032, 2000.

[184] A. A. suite. 2016.

[185] Y. Sun, M. Liu, and M. Q.-H. Meng. Wifi signal strength-based robot indoor localization. In *2014 IEEE International Conference on Information and Automation (ICIA)*, pages 250–256. IEEE, 2014.

[186] F. Tang, Z. M. Fadlullah, B. Mao, and N. Kato. An intelligent traffic load prediction-based adaptive channel assignment algorithm in sdn-iot: A deep learning approach. *IEEE Internet of Things Journal*, 5(6):5141–5154, 2018.

[187] T. C. Thang, Q.-D. Ho, J. W. Kang, and A. T. Pham. Adaptive streaming of audiovisual content using mpeg dash. *IEEE Transactions on Consumer Electronics*, 58(1), 2012.

[188] T. C. Thang, H. T. Le, A. T. Pham, and Y. M. Ro. An evaluation of bitrate adaptation methods for http live streaming. *IEEE Journal on Selected Areas in Communications*, 32(4):693–705, 2014.

[189] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf. 2006.

[190] S. Tomar. Converting video formats with ffmpeg. *Linux Journal*, 2006(146):10, 2006.

[191] R. Tönjes, P. Barnaghi, M. Ali, A. Mileo, M. Hauswirth, F. Ganz, S. Ganea, B. Kjærgaard, D. Kuemper, S. Nechifor, et al. Real time iot stream processing and large-scale data analytics for smart city applications. In *poster session, European Conference on Networks and Communications*. sn, 2014.

[192] S. Tozlu, M. Senel, W. Mao, and A. Keshavarzian. Wi-fi enabled sensors for internet of things: A practical approach. *IEEE Communications Magazine*, 50(6), 2012.

[193] A. L. Valdivieso Caraguay, A. Benito Peral, L. I. Barona Lopez, and L. J. Garcia Villalba. Sdn: Evolution and opportunities in the development iot applications. *International Journal of Distributed Sensor Networks*, 10(5):735142, 2014.

[194] M. Vardhana, N. Arunkumar, E. Abdulhay, and P. Vishnuprasad. Iot based real time trafic control using cloud computing. *Cluster Computing*, pages 1–10, 2018.

[195] A. Varga and R. Hornig. An Overview of the OMNeT++ Simulation Environment. In *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[196] A. Vega, C.-C. Lin, K. Swaminathan, A. Buyuktosunoglu, S. Pankanti, and P. Bose. Resilient, uav-embedded real-time computing. In *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pages 736–739. IEEE, 2015.

[197] C. Veness. Calculate Distance and Bearing between two Latitude/Longitude Points using Haversine Formula in JavaScript. *Movable Type Scripts*, 2011.

[198] T. Verbelen, P. Simoens, F. De Turck, and B. Dhoedt. Cloudlets: Bringing the Cloud to the Mobile User. In *Proceedings of the third ACM workshop on Mobile cloud computing and services*, pages 29–36. ACM, 2012.

[199] L. Verma, M. Fakharzadeh, and S. Choi. Wifi on steroids: 802.11 ac and 802.11 ad. *IEEE Wireless Communications*, 20(6):30–35, 2013.

[200] R. Vilalta, A. Mayoral, D. Pubill, R. Casellas, R. Martínez, J. Serra, C. Verikoukis, and R. Muñoz. End-to-end sdn orchestration of iot services using an sdn/nfv-enabled edge node. In *2016 Optical Fiber Communications Conference and Exhibition (OFC)*, pages 1–3. IEEE, 2016.

[201] Y. Wen-Bin, M. Souryal, and D. Griffith. LTE uplink performance with interference from in-band device-to-device (D2D) communications. In *IEEE Wireless Communications and Networking Conference*, pages 669–674, March 2015.

[202] D. Weyns, M. U. Iftikhar, D. Hughes, and N. Matthys. Applying architecture-based adaptation to automate the management of internet-of-things. In *Software Architecture - 12th European Conference on Software Architecture, ECSA 2018, Madrid, Spain, September 24-28, 2018, Proceedings*, pages 49–67, 2018.

[203] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra. Overview of the h. 264/avc video coding standard. *IEEE Transactions on circuits and systems for video technology*, 13(7):560–576, 2003.

[204] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley. Design, implementation and evaluation of congestion control for multipath tcp. In *NSDI*, volume 11, pages 8–8, 2011.

[205] Q. Wu, Y. Zeng, and R. Zhang. Joint Trajectory and Communication Design for multi-UAV enabled wireless Networks. *IEEE Transactions on Wireless Communications*, 17(3):2109–2121, 2018.

[206] Z. Xiao, P. Xia, and X.-G. Xia. Enabling uav cellular with millimeter-wave communication: Potentials and approaches. *IEEE Communications Magazine*, 54(5):66–73, 2016.

[207] X. Xie, X. Zhang, S. Kumar, and L. E. Li. pistream: Physical layer informed adaptive video streaming over lte. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking*, pages 413–425. ACM, 2015.

[208] N. Xiong and P. Svensson. Multi-sensor management for information fusion: issues and approaches. *Information fusion*, 3(2):163–186, 2002.

[209] K.-W. Yip and T.-S. Ng. A simulation model for nakagami-m fading channels, m¡ 1. *IEEE Transactions on Communications*, 48(2):214–221, 2000.

[210] C. Yu, O. Tirkkonen, K. Doppler, and C. Ribeiro. On the performance of device-to-device underlay communication with simple power control. In *IEEE 69th Vehicular Technology Conference*, pages 1–5, 2009.

[211] X. Yuan, J. Yang, P. Llull, X. Liao, G. Sapiro, D. J. Brady, and L. Carin. Adaptive temporal compressive sensing for video. In *2013 IEEE International Conference on Image Processing*, pages 14–18. IEEE, 2013.

[212] A. Zambelli. Iis smooth streaming technical overview. *Microsoft Corporation*, 3:40, 2009.

[213] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi. Internet of things for smart cities. *IEEE Internet of Things Journal*, 1(1):22–32, 2014.

[214] N. R. Zema, A. Trotta, G. Sanahuja, E. Natalizio, M. Di Felice, and L. Bononi. CUS-CUS: an Integrated Simulation Architecture for Distributed Networked Control Systems. In *14th IEEE Annual Consumer Communications & Networking Conference (CCNC), 2017*, pages 287–292. IEEE, 2017.

[215] Y. Zeng, J. Lyu, and R. Zhang. Cellular-connected uav: Potential, challenges, and promising technologies. *IEEE Wireless Communications*, 26(1):120–127, 2019.

[216] R. Zhang, M. Wang, L. X. Cai, Z. Zheng, X. Shen, and L.-L. Xie. Lte-unlicensed: the future of spectrum aggregation for cellular networks. *IEEE Wireless Communications*, 22(3):150–159, 2015.

[217] Z. Zhang, R. Q. Hu, Y. Qian, and A. Papathanassiou. D2D communication underlay in uplink cellular networks with fractional power control and fractional frequency reuse. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7, 2015.

[218] Y. Zhao, M. Song, and C. Xin. Fmac: A fair mac protocol for coexisting cognitive radio networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 1474–1482. IEEE, 2013.

[219] Z. Zhou, Z. Yang, C. Wu, W. Sun, and Y. Liu. Lifi: Line-of-sight identification with wifi. In *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pages 2688–2696. IEEE, 2014.

[220] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, volume 2, pages 28–31. IEEE, 2004.

[221] L. Ziyang, P. Tao, X. Shangwen, and W. Wang. Mode selection for device-to-device (D2D) communication under LTE-advanced networks. In *IEEE International Conference on Communications*, pages 5563–5567, June 2012.