UNIVERSIDAD CARLOS III DE MADRID

TESIS DOCTORAL

PREDICTION-BASED TECHNIQUES FOR THE OPTIMIZATION OF MOBILE NETWORKS

Autor: Nicola Bui, University Carlos III of Madrid
IMDEA Networks Institute
Director: Joerg Widmer, IMDEA Networks Institute

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

Leganés (Madrid), Mayo de 2017

UNIVERSIDAD CARLOS III DE MADRID

PH.D. THESIS

PREDICTION-BASED TECHNIQUES FOR THE OPTIMIZATION OF MOBILE NETWORKS

Author: Nicola Bui, University Carlos III of Madrid
IMDEA Networks Institute
Director: Joerg Widmer, IMDEA Networks Institute

DEPARTMENT OF TELEMATIC ENGINEERING

Leganés (Madrid), May 2017

Prediction-based Techniques for the Optimization of Mobile Networks

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Prepared by
Nicola Bui, University Carlos III of Madrid IMDEA Networks Institute

Under the advice of
Joerg Widmer, IMDEA Networks Institute
Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid

---

Date: Mayo, 2017

Web/contact: nicola.bui@imdea.org

This work has been supported by IMDEA Networks Institute.

TESIS DOCTORAL

Prediction-based Techniques for the Optimization of Mobile Networks

Autor: Nicola Bui, University Carlos III of Madrid
IMDEA Networks Institute
Director: Joerg Widmer, IMDEA Networks Institute

Firma del tribunal calificador:

Calificación:

Leganés, de de

Acknowledgements

My first and foremost thank goes to my advisor, Joerg Widmer, who has always been there when I needed the most and always believed in me even and in particular when I doubted myself. For this and for his ability to seamlessly alternate between opposite roles such as the rigorous reviewer and the helpful collaborator, I cannot think of a better guide for a PhD student.

My second thought goes to my PhD fellows and colleagues at IMDEA Networks, who never failed to help when I need assistance and was always present to share a coffee along with some very insightful or simpler light-hearted chats.

My last thank goes to my family and my wife, Claudia Boccato, above all. She has been my safe heaven where to return after the bad days and was always there to celebrate my, modest, successes.

Abstract

Mobile cellular networks are complex system whose behavior is characterized by the superposition of several random phenomena, most of which, related to human activities, such as mobility, communications and network usage. However, when observed in their totality, the many individual components merge into more deterministic patterns and trends start to be identifiable and predictable.

In this thesis we analyze a recent branch of network optimization that is commonly referred to as anticipatory networking and that entails the combination of prediction solutions and network optimization schemes. The main intuition behind anticipatory networking is that knowing in advance what is going on in the network can help understanding potentially severe problems and mitigate their impact by applying solution when they are still in their initial states. Conversely, network forecast might also indicate a future improvement in the overall network condition (i.e. load reduction or better signal quality reported from users). In such a case, resources can be assigned more sparingly requiring users to rely on buffered information while waiting for the better condition when it will be more convenient to grant more resources.

In the beginning of this thesis we will survey the current anticipatory networking panorama and the many prediction and optimization solutions proposed so far. In the main body of the work, we will propose our novel solutions to the problem, the tools and methodologies we designed to evaluate them and to perform a real world evaluation of our schemes.

By the end of this work it will be clear that not only is anticipatory networking a very promising theoretical framework, but also that it is feasible and it can deliver substantial benefit to current and next generation mobile networks. In fact, with both our theoretical and practical results we show evidences that more than one third of the resources can be saved and even larger gain can be achieved for data rate enhancements.

Table of Contents

List of Tables

List of Figures

List of Acronyms

ACK	Acknowledgment
ANN	Artificial Neural Network
AR	AutoRegressive
ARC	Admission and Resource Control
ARIMA	AutoRegressive Integrated Moving Average
ARMA	AutoRegressive Moving Average
ASU	Arbitrary Strength Unit
BS	Base Station
C-RNTI	Cell RNTI
CCN	Content Centric Network
CDF	Cumulative Distribution Function
CF	Collaborative Filtering
CFI	Control Format Indicator
ConvOpt	Convex Optimization
CRC	Cyclic Redundancy Check
CR	Cognitive Radio
CSI	Channel State Information
CTM	Continuous Time Markov
CTMC	Continuous Time Markov Chain
D2D	device-to-device

DASH	Dynamic Adaptive Streaming over HTTP
DCI	Downlink Control Information
DiffServ	Differentiated Services
DTMC	Discrete Time Markov Chain
eCDF	empirical Cumulative Distribution Function
ELM	Extreme Learning Machine
epdf	empirical probability density function
FARIMA	AutoRegressive Fractionally Integrated Moving Average
FDD	Frequency Division Multiplexing
FTP	File Transfer Protocol
GARCH	Generalized AutoRegressive Conditionally Heteroskedastic
GP	Gaussian Process
GPS	Global Positioning System
HMM	Hidden Markov Models
HLS	HTTP Live Streaming
HSPA	High Speed Packet Access
HSDPA	High Speed Downlink Packet Access
HTTP	Hypertext Transfer Protocol
ID	identity
ILP	Integer Linear Programming
IP	Internet Protocol
KKF	Kriged Kalman Filter
KPI	Key Performance Indicator
LTE	Long Term Evolution
LP	Linear Programming
LZ	Lempel-Ziv

MA	Moving Average
MC	Markov Chain
MCS	Modulation and Coding Scheme
MIB	Master Information Block
MILP	Mixed-Integer Linear Programming
MNLP	Mixed Non-Linear Program
MPC	Model Predictive Control
MDP	Markov Decision Process
MRA	MultiResolution Analysis
MSE	Mean Square Error
NIC	Networks Interface Card
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
OWL	Online Watcher for LTE
PCI	Physical Cell ID
PDCCH	Physical Downlink Control Channel
PDSCH	Physical Downlink Shared Channel
PF	Proportionally Fair
PSS	Primary Synchornization Sequence
QoE	Quality-of-Experience
QoS	Quality-of-Service
RA	Resource Allocation
RA-RNTI	Random Access RNTI
RAN	Radio Access Network
RAR	Random Access Response
RB	Resource Block

RE	Resource Element
REM	Radio Environment Map
RNTI	Radio Network Temporary Identifier
RRC	Radio Resource Control
SDR	Software Defined Radio
SFN	System Frame Number
SINR	Signal-to-Interference-plus-Noise Ratio
SSS	Secondary Synchronization Sequence
SVM	Support Vector Machine
TB	Transport Block
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTI	Transmission Time Interval
UDP	User Datagram Protocol
UE	User Equipment

Part I :   Introduction

Chapter 1

Summary

"Every form of behavior is compatible with determinism. One dynamic system might fall into a basin of attraction and wind up at a fixed point, whereas another exhibits chaotic behavior indefinitely, but both are completely deterministic" [1]. In his provocative essay, Ted Chiang is suggesting that unpredictability is just a consequence of the limitedness of human comprehension. Without going as far as calling mobile networks deterministic phenomena, in this thesis we investigate how to predict them and how to exploit prediction for their optimization.

Anticipatory networking is a recent branch of wireless communications where performance optimization is driven by prediction of the future evolution of the system. In such a way, communication systems can exploit different context information to evaluate the opportunity of making a given decision not only in terms of the current system state, but also accounting for what is going to happen in the system. The main idea is that if we are able to predict the system state evolution and possible disruptive events, we can make better decisions and take actions directed to steer the system so that not only can we mitigate the impact of serious treats, but also we can exploit indicators of forthcoming improvements to adopt more uncompromising policies.

For instance, a common example for disruptive events is the impact of a large crowd meeting at a given location: in such an event a few network cells need to face a much larger traffic than what their usual level is. However, these events are very rarely abrupt. Usually, indicators of such a situation can be seen well in advance: for example, the number of connected users is larger than the average and growing and, if mobility information is available, the density of people in surrounding areas can hint to many persons moving towards the event center. These indicators, together with more direct ones, such as an obvious traffic increase, and more subtle ones, such as twitter messages originated in the area with similar contents, can be combined together to identify a possible network issue. In such a case, the more serious effects of the event can be counteracted by adopting more conservative load balancing policies that refrain from direct additional traffic to the area, or more careful resource allocation and admission control strategies can be selected to avoid outage or, at least, limit it.

Moreover, anticipatory networking does not only focus on large scale disruptive events, but also it considers more nuanced variation in the system conditions both as perceived by the system as a whole and by single users. For instance, the signal quality perceived by a mobile user moving between different cells depends on many factors. The first of which is the pathloss, i.e. the signal degradation due to the distance between the user's mobile and the base station antenna; then the surrounding environment contributes to it in form of fading, i.e., the attenuation variations due, for instance, to multi-path propagation, and other factors such as how many other users are present and the traffic they may produce. All these parameters can be monitored and, with different level of accuracy, predicted. Therefore, the user's mobile phone (or some control software on it) can tune its operations according to current and expected requirements and conditions. Alternatively, the optimization can be driven by network operators by tuning the resource allocation of all users connected to the same base station taking into account their individual predictions, their mutual relationships and the conditions of the neighboring cells.

In particular, the most straightforward use case, which is also used to evaluate the solutions proposed in the rest of this thesis, is multimedia streaming. Multimedia streaming is one of the main source of mobile traffic [2] and its characteristics make it the perfect application for anticipatory networking evaluation: streaming applications use buffer to avoid outages, they present data requirements variable with time and that may be varied according to the desired quality. Anticipatory networking solutions can exploit the buffer to decide when to request more resources from the network (i.e. when the signal quality is higher and communication is more efficient) and when to rely on buffered content to avoid consuming resources (i.e. bad signal quality).

In this thesis, we analyze prediction-based techniques for the optimization of mobile networks, by also addressing practical aspects and providing a thorough evaluation based on both synthetic traces and actual measurements. Prediction-based optimization is one of the most archetypal formulation of anticipatory networking: it makes use of one or more forecasting techniques to drive the solution of a given optimization problem. In particular, in this thesis we mainly focus on ARIMA and statistical models for what concerns the prediction part and on convex optimization, linear programming and heuristics derived from both to solve the optimization part.

The rest of thesis is split into four parts that cover the following aspects of the work: Part I contains introductory material and discusses preliminary topics; Part II contains the main theoretical results; Part III is dedicated to practical aspects related to the realization of anticipatory networking solutions and Part IV summarizes the overall achievements by illustrating how the proposed techniques perform in real environment and compare them to theoretical bounds; this final part also provides the conclusions to this thesis. In the following, we provide brief summaries of each of the parts of the thesis together with a selection of the most representative results.

The rest of this introductory part is structured as follows: after a list of the published papers and their contributions to this thesis, Chapter 2 presents a comprehensive survey on anticipatory networking solutions for mobile networks, which is an extract of [3]. This survey presents a classification of anticipatory networking techniques based on the type of contextual information used to control the optimization framework. In addition, two brief handbooks are provided to analyze the possible alternatives in terms of prediction and optimization techniques. Chapter 3 and Chapter 4 present two models to characterize prediction errors when forecasting is applied to estimate the achievable data rate of mobile users. In particular, the former presents a two-part framework to deal with short- and medium- long-term predictions, while the latter proposes a heuristic to estimate the short-term prediction errors based on Gaussian random walks.

The second part of the thesis will present our original optimization solutions and is split into three chapters. Chapter 5 describes our first anticipatory networking technique, which combines an optimal solution to the omniscient resource allocation minimization problem with the two-scale error model of Chapter 3 to account for prediction uncertainty. Chapter 6 extends the optimization problem to the multi-user case and accounts for quality maximization when all the users are served with minimum outage: methods to compute the optimal solution and a fast heuristic are provided. Chapter 7 introduces guaranteed quality of service in the optimization framework: in particular, the proposed solution computes the largest set of users that can be served by the system guaranteeing the specified quality to each of them. This initial solutions are validated on synthetic datasets and show the potential benefits of anticipatory networking solutions: theoretical resource savings quality improvements of about 50% have been measured in our dataset when perfect prediction has been used, while the performance of the heuristic solutions were still improving over the baseline, but worse than the optimal of about 20%.

The third part of the thesis describes the tools and methodologies we developed to evaluate the practical performance of our solutions. The main tool we designed and developed is a decoder of the LTE control channel and is described in Chapter 8: thanks to a software defined radio and our software it is possible to collect uplink and downlink information about LTE scheduling (i.e. modulation and coding scheme used by the different users, their assigned resource and their actual data rate) on a per millisecond granularity. The tool is publicly available at: `https://git.networks.imdea.org/nicola_bui/imdeaowl`. First, we evaluated how effectively mobile phones can estimate the achievable data rate obtainable from an LTE cell. Subsequently, we started collecting a month-worth of data in four locations in Madrid and Leganes on which evaluate our solutions in details. Chapter 9 evaluates how accurate is the estimation of mobile phones achievable rate by using data collected with my sniffer. We concluded that accurate and precise measurements can be obtained with mobile phones even in case of short-lived communications (e.g. 50 ms or 100 KBytes). However, different phones present different biases. An additional contribution, which provides more details on a lightweight methodology to estimate achievable data link rates using mobile phone applications is provided in Appendix A.

The fourth and last part of this thesis will present our practical validation of anticipatory networking solutions in Chapter 10 and will summarize our conclusions in Chapter 11. In particular, Chapter 10 will present and analyze the dataset collected using my LTE sniffer in terms of predictability and possible benefit obtainable from anticipatory networking solutions. We analyzed the four datasets (one per location) both as aggregated cell information and as individual user statistics. After showing the main characteristics of the datasets we applied a comprehensive optimization framework to evaluate different resource allocation techniques coupled with variable forecasting accuracy. In particular, we evaluated 1) the fraction of saved network resources keeping constant the quantity of data offered to each users and 2) the data rate improvement keeping constant the quantity of resources assigned to each users. Not only did the analysis on the real datasets validate the theoretical results, but we measured performance improvements even larger than the ones obtained on synthetic traces. This fact is mainly due to the way different users were generated for the synthetic traces: by picking them from the same statistical distribution does not re-create the same level of variety that is present in real datasets. In turn, the differences among users allowed the optimization framework to obtain better results. With average resource savings of about 40% and about twice the data rate, we believe anticipatory networking is a feasible solution for current and even more for the networks of the next generation.

1.1. Contributions

The main body of the thesis comprehends 13 publications, of which 3 are journals (one has just been submitted), 7 are conference papers (one of which is still under review), 1 is a minor workshop and 2 are two demo papers. The ranking specified for conference papers is based on either the CORE2014 or the ERA2010 datasets (see `http://portal.core.edu.au` and `http://www.conferenceranks.com/`), while for journals the Journal Citation Reports (JCR) percentile and quartile is specified (see `https://jcr.incites.thomsonreuters.com/`). In all the papers where I am not the first author, I specified what was my contribution to the work.

In details,

- The prediction error model is published in [4] (conference, first author, rank N/A).
- The link between prediction error and Gaussian random walks is published in [5] (workshop paper, first author, rank N/A).
- The survey on anticipatory networking techniques is published in [3] (journal, first author, JCR 99.653 - Q1).
- The resource allocation solution for single user which accounts for prediction errors is published in [6] (conference, first author, rank A).
- The multi-user variable quality heuristic for multi-objective resource allocation is published in [7] (conference, first author, N/A).
- The admission control extension of the multi-user variable quality solution is published in [8] (conference, first author, rank A).

- The optimization and prediction techniques comprehensive framework is currently under review in [9] (conference, first author, rank A) and is being extended as a journal contribution for [10] (journal, first author, JCR 87.847 - Q1).

- The description of the LTE sniffer is published in [11] (conference, first author, N/A).

- The work on mobile phone lightweight measurements is published in [12] (conference, second author, rank A) and in [13] (journal, second author, JCR 80.903 - Q1): I mainly contributed to this work providing the methods to estimate the achievable data rate from the packet dispersion rate and how to implement it on the measurements.

- Two demo papers has been published as proof-of-concepts of the thesis [14] (conference, second author, rank B) and [15] (conference, third author, rank B): both demos were a joint work with a German group who presented the demonstration.

Besides the core of the thesis, I published 3 other journals and 2 other conference papers. In details,

- In collaboration with TU Darmstadt I contributed in two works that evaluated the quality-of-service of mobile networks thanks to crowd-sourced data. These works are published in [16] (conference, third author, rank B) and [17] (journal, third author, JCR 55.903 - Q2)

- I contributed to a work on LTE energy saving policies [18] (conference, third author, rank B), which was lead by the Center of Technology and Telecommunications of Catalunya: here I provided baseline policies and I contributed to obtaining the final results.

- Two journals that I started before starting the PhD program have been published in the PhD period. The first describes a feasible solution for urban environmental monitoring based on low-power devices [19] (journal, second author, JCR N/A): for this work I was in charge of the design of SW and HW realization of the low power devices used in the project. The second describes a nested optimization framework for self-sustainable low-power devices [20] (journal, first author, JCR 62.153 - Q2). Both works are in collaboration with the University of Padova.

Chapter 2

Survey of Anticipatory Networking Solutions for Mobile Networks

This chapter investigates anticipatory networking, a recent research direction that supports network optimization through system state prediction. Anticipatory networking is enabled by prediction tools and the recent and huge increase in data availability. In addition, data centers are becoming more and more important in providing services and tools to access and analyze huge amounts of data.

This chapter reviews the recent literature of prediction-based solutions that are proposed for wireless mobile networks. In addition, this survey delves into the following questions: How can prediction support wireless networks? Which type of information is possible to predict and which applications can take advantage of it? Which tools are the best for a given scenario or application? Which scenarios, among the ones envisioned for 5G networks, can benefit the most from anticipatory networking? What is yet to be studied in order for anticipatory networking to be implemented in 5G networks?

A typical anticipatory networking solution is usually characterized by the following three attributes, which also determine the structure of this survey:

- *Context* defines the type of information considered to forecast the system evolution.
- *Prediction* specifies how the system evolution is forecast from the context.
- *Optimization* describes how prediction is exploited to meet the application objectives.

With reference to a typical access control problem, the anticipatory networking solution might exploit the history of Global Positioning System (GPS) information (*context*) to train an AutoRegressive (AR) model (*prediction* part) to predict the future positions of the users and their channel conditions to solve an Integer Linear Programming (ILP) problem (*optimization* part) that maximizes their Quality-of-Experience (QoE).

We split the main body of the anticipatory networking literature into four categories based on the context used to characterize the system state and to determine its evolution: *geographic*, in the form of human mobility patterns derived from location-based information; *link*, coded as channel gain, noise and interference levels obtained from reference signal feedback; *traffic*, represented by network load, throughput, and occupied physical resource blocks based on higher-layer performance indicators; *social*, such as user's behavior, profile, and information derived from user-generated contents and social networks.

In order to determine which techniques are the most suitable to solve a given problem, it is important to analyze the following:

- *Properties* of the context:

1) *Dimension* describes the number of variables predicted by the model, which can be uni- or multivariate.
2) *Granularity and precision* define the smallest variation of the parameter considered by the context and the accuracy of the data: the lower the granularity, the higher the precision and vice versa. Temporal and spatial granularities are crucial to strike a balance between efficiency and accuracy.
3) *Range* characterizes the distance (usually time or space) between known data samples and the farthest predicted sample. It is also known as prediction (or optimization) horizon.

- *Constraints* of the prediction or optimization model:

1) *Availability of physical model* states whether a closed-form expression exists to describe the phenomenon.
2) *Linearity* expresses the quality of the functions linking inputs and outputs of a problem.
3) *Side information* determines if the context depends on auxiliary information.
4) *Reliability and validity of information* specifies the noisiness of the data set, depending on which the prediction robustness should be calibrated.

Table 2.1: Survey classification and structure

Context (Section 2.1)		**Prediction** (Section 2.2)	**Optimization** (Section 2.3)
	Geo	*Ideal:* [21–24]	*ConvOpt*[a]*:* [22]
		Time series: [4,25–30]	*MDP*[b]*/MPC*[c]*:* [31,32]
		Regression, classification: [33–40]	*Game theory:* [41]
		Probabilistic: [31,32,42–51]	*Heuristic:* [21,23,29,30,38,39,51]
	Link	*Ideal:* [7,8,52–65]	*ConvOpt:* [7,8,52–57,60–66]
		Time series: [6,67–69]	*MDP/MPC:* [70–73]
		Regression, classification: [66,74–79]	*Game theory:* [80]
		Probabilistic: [70–73,81–83]	*Heuristic:* [6,67–69,78,79,82,83]
	Traffic	*Ideal:* [84–91]	*ConvOpt:* [85,87,91–98]
		Time series: [93,99–105]	*MDP/MPC:* [90,99,105,106]
		Regression, classification: [94–97,107–115]	*Game theory:* [116]
		Probabilistic: [106,112,117]	*Heuristic:* [84,86,89,108,111,113,114]
	Social	*Ideal:* [118–121]	*ConvOpt:* [119,120,122–124]
		Time series: [125]	*MDP/MPC:* [126]
		Regression, classification: [127–133]	*Game theory:* [41,80,116,134,135]
		Probabilistic: [80,116,122–124,126,136–139]	*Heuristic:* [118,121,125,127–130,138,139]

Surveys: [140–159]

Projects: [160–162]

[a] convext optimization [b] Markov decision process [c] model predictive control

The classification section provide the reader with the link between the different contexts and the solutions adopted to satisfy the given application requirements. Also, this section is meant to illustrate the range of anticipatory networking solutions. The two handbooks that follow have the twofold objective of reviewing the tools adopted in the literature and to analyze them in terms of variables of interest and constraints of the models.

Table 2.1 provides a mapping between the techniques described in Section 2.2 and 2.3 (columns) and the context discussed in Section 2.1 (rows). Each main category is further split into subcategories according to its internal structure. Namely, the prediction category is subdivided into ideal (perfect prediction is assumed to be available), time series predictive modeling, similarity-based classification and regression analysis, and probabilistic methods. The optimization category is split into Convex Optimization (ConvOpt), Markov Decision Process (MDP) and Model Predictive Control (MPC), game theoretic and, heuristic approaches.

After the analytical part, Section 2.4 concludes the survey discussing the impact of anticipatory networking on future networks, the envisioned hindrances to its implementation and the next open challenges.

2.1. Context-Based Classification

This section illustrates the different types of context that can be predicted and exploited. Each context is illustrated by highlighting the most popular prediction techniques as well as the applications enhanced by anticipatory optimization.

2.1.1. Geographic Context

Geographic context refers to the geographic area associated with a specific event or information. For what concerns wireless communications, the geographic context is usually the mobile user's location enriched with speed information as well as past and future trajectories. Fig. 2.1 illustrates an example of estimated trajectories of 6 mobile users.

User mobility is shown to have a predictability as high as 93% [42], at least for a high-income country with stable social conditions. Similarly, [49] investigates both the maximal predictability and how close to this value practical algorithms can come when applied to a large mobile phone dataset. Those results indicate that human mobility is very far from being random. Therefore, collecting, predicting and exploiting geographic context is of crucial importance.

The rest of this section is organized according by the main focus of the reviewed papers: the two major groups of them deals with pure geographical prediction, and multimedia streaming optimization, respectively.

2.1.1.1. Next location prediction

The simplest approach is to forecast where a given user will be at a predetermined instant of time in the future. The authors of [27] propose to track mobile nodes using topological coordinates

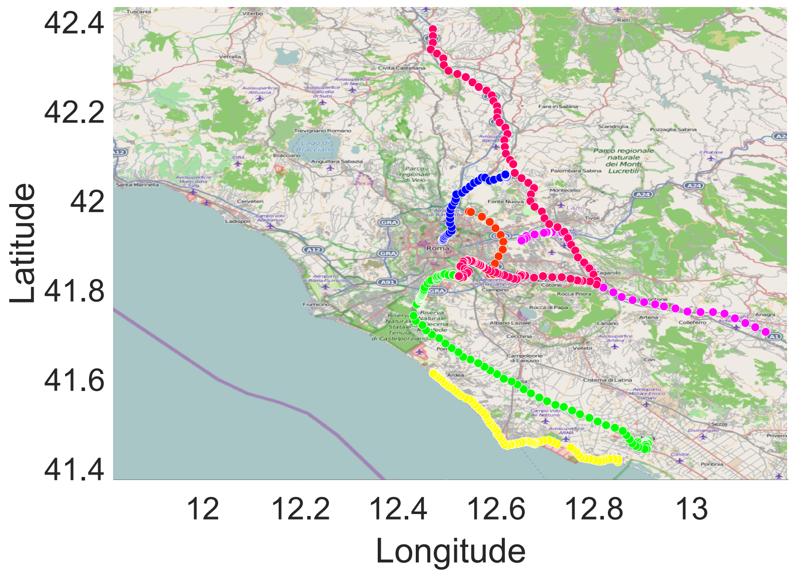


Figure 2.1: Geographic context example: an example of estimated trajectories of 6 mobile users.

and topology preserving maps. Nodes' location is identified with a vector of distances (in hops) from a set of nodes called anchors and a linear predictor is used to estimate the mobile nodes' future positions. Evaluation is performed on synthetic data and nodes are assumed to move at constant speed. Results show that the proposed method approaches an accuracy above 90% for a prediction horizon of some tens of seconds.

A more general approach that exploits Artificial Neural Networks (ANNs) is discussed in [37]. Extreme Learning Machines (ELMs), which do not require any parameter tuning, are used to speed up the learning process. The method is evaluated using synthetic data over different mobility models.

Location information can be extracted from cellular network records. In this way the granularity of the prediction is more coarse, but positioning can be obtained with little extra energy. In particular, [50] aims at predicting a given user location from those of similar users. *Collective behavioral patterns* and a Markovian predictor are used to compute the next six locations of a user with a one-hour granularity, i.e., a six-hour prediction horizon. Evaluation is done using a real dataset and shows that an accuracy of about 70% can be achieved in the first hour, decreasing to $40 - 50\%$ for the sixth hour of prediction.

Users' locations and short-term trajectories are exploited to extend the prediction horizon [33] and, in turn, to predict the next handover. The authors use Channel State Information (CSI) and handover history to solve a classification problem via supervised learning, i.e., employing a multi-class Support Vector Machine (SVM). In particular, each classifier corresponds to a possible previous cell and predicts the next cell. A real-time prediction scheme is proposed and the feedback is used to improve the accuracy over time. Simulation results have been derived using both synthetic and real datasets.

2.1.1.2. Space and time prediction

Statistical models are often used to describe the prediction of mobility in a combined space-time domain. In [43], the idea is to predict not only the future location a user will reach, but also *when* and for *how long* the user will stay there. Mobility is modeled as a semi-Markov process to incorporate the *sojourn* time during which a user remains in a certain location. In particular, the transition probability matrix and the sojourn time distribution are derived from the previous association history. A similar approach is presented in [44], which extends prediction from single to multi-transitions. Both papers provide also some preliminary results on the benefits of the prediction on resource allocation and balancing.

The interdependence between time and space is investigated also in [34] by examining real data collected from smartphones during a two month deployment. Furthermore, [47] shows the benefit of using a location-dependent Markov predictor with respect to a location-independent model based on nonlinear time series analysis. Additionally, it is shown that information on arrival times and periodicity of location visits is needed to provide accurate prediction. A system design, named SmartDC, is presented in [31, 32, 51]. SmartDC comprises a mobility learner, a mobility predictor and an adaptive duty cycling. The proposed location monitoring scheme optimizes the sensing interval for a given energy budget. The system has been implemented and tested in a real environment. Notably, this is also one of the few papers that takes into account the *cost* of prediction, which in this case is evaluated in terms of energy. Namely, the authors detect approximately 90% of location changes, while reducing energy consumption at the expense of higher detection delay.

In [45], the authors represent the network coverage and movements using graph theory. The user mobility is modeled using a Continuous Time Markov (CTM) process where the prediction of the next node to be visited depends not only on the current node but also on the previous one (i.e., second-order Markovian predictor). Considering both local as well as global users' profiles, [46] extends the previous Markovian predictor and improves accuracy by about 30%. As pointed out in [48], sojourn times and transition probabilities are inhomogeneous. Thus, an inhomogeneous CTM process is exploited to predict user mobility. Evaluation on a real dataset shows an accuracy of 67% for long time scale prediction.

2.1.1.3. Location sequences and trajectories

Spatio-temporal prediction can encompass sequences of locations and users' trajectories. User mobility profiles have been introduced in [163] to optimize call admission control, resource management and location updates. Statistical predictors are used to forecast the next cell a mobile phone is going to connect to. The validation of the solution is carried on by simulation. Location prediction based on nonlinear time series analysis is presented in [25]. The framework focuses on the *temporal* predictability of users' location, considering their arrival and residence times in relevant places. The evaluation is done considering four different real datasets. The authors evaluate first the predictability of the considered data and then show that the proposed nonlinear predictor outperforms both linear and Markov-based predictors. Precision approaches $70 - 90\%$ for medium scale prediction (5 minutes) and decreases to $20 - 40\%$ for long scale (up to 8 hours).

Trajectory analysis and prediction also benefit from exploiting specific constraints such as streets, roads, traffic lights and public transportation routes. In [83] the authors adapt the local Markovian prediction model for a specific coverage area in terms of a set of roads, moving directions, and traffic densities. When applying Markov prediction schemes, the authors consider a road compression approach to avoid dealing with a large number of locations, reduce the size of the state space, and minimize the approximation error. A more attractive candidate for trajectory prediction is the public transportation system, because of known routes and stops, and the large amount of generated mobile data traffic. In [24], the authors investigate the predictability of mobility and signal variations along public transportation routes, to examine the viability of predictive content delivery. The analysis on a real dataset of a bus route, covering both urban and sub-urban areas, shows that modeling prediction uncertainty is paramount due to the high variability observed, which depends on combined effects of geographical area, time, forecasting window and contextual factors such as signal lights and bus stops.

In order to improve the accuracy of time series techniques, in [26] the authors exploit the movement of friends, people, and, in general, entities, with correlated mobility patterns. By means of multivariate nonlinear time series prediction techniques, they show that forecasting accuracy approaches 95% for medium time scale prediction (5 to 10 minutes) and is approximately 50% for 3 hour prediction. Confidence bands show a significant improvement when prediction exploits patterns with high correlation. Evaluation is done considering two different real datasets.

Kalman filtering is used to predict the future velocity and moving trends of vehicles and to improve the performance of broadcasting [30] when it comes for continuous trajectories. The main idea is that each node should send the message to be broadcast to the fastest candidate based on its neighbors' future mobility. Simulation results show modest gains, in terms of percentage of packet delivery and end-to-end delay, with respect to non-predictive methods.

Regression techniques [40] can be used instead of Kalman filters to analyze GPS observations of past trips. A systematic methodology, based on geometrical structures and data-mining techniques, is proposed to extract meaningful information for location patterns. This work characterizes the location patterns, i.e., the set of locations visited, for several millions of users using nationwide call data records. The analysis highlights statistical properties of the typical covered area and route, such as its size, average length and spatial correlation.

Similarly, [35] shows how the regularity of driver's behavior can be exploited to predict the current end-to-end route. The prediction is done by exploiting clustering techniques and is evaluated on a real dataset. A further approach, named *WhereNext*, is proposed in [36]. This method forecasts the next location of a moving object using past movement patterns that are based on both spatial and temporal information. The prediction is done by building a decision tree, whose nodes are the regions frequently visited. It is then used to predict the future location of a moving object. Results are shown using a real dataset provided by the GeoPKDD project [161]. The authors show the trade-off between the fraction of predicted trajectories and the accuracy. Both [35] and [36] show similar performance with an accuracy of approximately 40% and medium time scale prediction (order of minutes).

2.1.1.4. Dealing with errors

We analyzed the impact of estimation and prediction errors in [4]. We propose a comprehensive overview of several mobility predictors and associated errors and investigate the main error sources and their impact on prediction. Based on this, they propose a stochastic model to predict user throughput that accounts for uncertainty. The method is evaluated using synthetic data while assuming that prediction's errors have a truncated Gaussian distribution. The joint analysis on the predictability of location and signal strength, which in this case is simply quantified by the standard deviation of the random variable, shown in [24] indicates that location-awareness is a key factor to enable accurate signal strength predictions. Location errors are also considered in [28] where both temporal and spatial correlation are exploited to predict the average channel gain. The proposed method combines an AR model with functional linear regression and relies on location information. Results are derived using real data taken from the MOMENTUM project [160] and show that the proposed method outperforms SVM and AR processes.

2.1.1.5. Mobility-assisted handover optimization

Efficient resource reservation and context transfer procedures during handover are key to provide users with seamless mobility. To guarantee the service continuity for mobile users, the conventional in-advance resource reservation schemes make a bandwidth reservation over all the cells that a mobile host will visit during its active connection. By predicting mobility pattern, it is possible to prepare resources in the most probable cells for the moving users. Using a Markov chain-based pattern prediction scheme, the authors in [83] propose a statistical bandwidth management algorithm to handle proactive resource reservations to reduce bandwidth waste. Along similar lines, [45, 125] investigate mobility prediction schemes, also considering user profiles, time-of-day, and duration characteristics, to improve the handover performance in terms of resource utilization, handover accuracy, call dropping and call blocking probabilities.

2.1.1.6. Geographically-assisted video optimization

One of the most popular applications that clearly benefits from geographic context prediction is video streaming. A pioneer work showing the benefit of a long-term location-based scheduling for streaming is [29]. The authors propose a system for bandwidth prediction based on geographic location and past network conditions. Specifically, the streaming device can use a GPS-based bandwidth-lookup service in order to predict the expected bandwidth availability and to optimally schedule the video playout. The authors present simulation as well as experimental results, where the prediction is performed for the upcoming 100 meters. The predictive algorithm reduces the number of buffer underruns and provides stable video quality.

Application-layer video optimization based on prediction of user's mobility and expected capacity, is proposed also in [21, 22, 39]. In [21], the authors minimize a utility function based on system utilization and rebuffering time. For the single user case they propose an online scheme based on partial knowledge, whereas the multiuser case is studied assuming complete future knowledge. In [22], different types of traffic are considered: full buffer, file download and buffered video. Prediction is assumed to be available and accurate over a limited time window. Three different utility functions are compared: maximization of the network throughput, maximization of the minimum user throughput, and minimization of the degradations of buffered video streams. Both works show results using synthetic data and assuming perfect prediction of the future wireless capacity variations over a time window with size ranging from tens to hundreds of seconds. In contrast, [39] introduces a data rate prediction mechanism that exploits mobility information and is used by an enhanced Proportionally Fair (PF) scheduler. The performance gain is evaluated using a real dataset and shows a throughput increase of 15%-55%.

Delay tolerant traffic can also benefit from offloading and prefetching as shown in [23]. The authors propose methods to minimize the data transfer over a mobile network by increasing the traffic offloaded to WiFi hotspots. Three different algorithms are proposed for both delay tolerant and delay sensitive traffic. They are evaluated using empirical measurements and assuming errors in the prediction. Results show that offloaded traffic is maximized when using prediction, even when this is affected by errors.

A *geo-predictive streaming system* called GTube, is presented in [38]. The application obtains the user's GPS locations and informs a server which provides the expected connection quality for future locations. The streaming parameters are adjusted accordingly. In particular, two quality adaptation algorithms are presented, where the video quality level is adapted for the upcoming 1 and n steps, respectively, based on the estimated bandwidth. The system is tested using a real dataset and shows that accuracy reaches almost 90% for very short time scale prediction (few seconds), but it decreases very fast approaching zero for medium time scale prediction (few minutes). However, the proposed n-step algorithm improves the stability of the video quality and increases bandwidth utilization.

2.1.2. Link Context

We refer to Link context as the prediction of the physical wireless channel, i.e., the channel quality and its characteristics, so that it is possible either to take advantage of future link improvements or to counter bad conditions before they impact the system. As an example of link context, Fig. 2.2 shows a pathloss map of the center of Berlin realized with the data of the MOMENTUM [160] project.

2.1.2.1. Channel parameter prediction

One possible approach to anticipate the evolution of the physical channel state is to predict the specific parameters that characterize it. In general, both large- and small-scale fading is the cause of the variations of the physical channel. Although it might not be possible to predict fast fading, pathloss and shadowing effects have been shown to be predictable and has been the focus of several papers. In [79], the time-varying nonlinear wireless channel model is adopted to predict the channel quality variation anticipating distance and pathloss exponent. The performance evaluation is done using both an indoor and an outdoor testbed. The goodput obtained with the proposed bitrate control scheme can be almost doubled compared to other approaches.

The authors of [77] propose a two-step approach that combines machine learning and dimensional reduction techniques. Specifically, they propose a new model for generating the input vector, the dimension of which is reduced by applying linear and nonlinear principal component analysis. A trained learning machine is then use to process the reduced vector. The authors compare ANNs and SVMs using real measurements and conclude that slightly better results can be achieved using the ANN regressors.

Supporting the temporal prediction with spatial information is proposed in, e.g., [74] to study the evolution of shadow fading. To this extent a Kriged Kalman Filter (KKF) is proposed to track the time varying shadowing using a network of Cognitive Radios (CRs). The prediction is used to anticipate the position of the primary users and the expected interference and, consequently, to maximize the transmission rate of CR networks. Errors with the proposed model approach 2 dB (compared to 10 dB obtained with the pathloss based model). A similar objective is aimed at in [72], which formulates the CR throughput optimization problem as an MDP. In particular, the predicted channel availability is used to maximize the throughput and to reduce the time overhead of channel sensing. Predictors robust to channel variations are investigated also in [78]. A clustering method with supervised SVM classification is proposed. The performance is shown for bulk data transport via Transmission Control Protocol (TCP) and it is also shown that the predictive approach outperforms non-predictive ones.

Finally, maps are a popular tool to provide geo-referenced predictions; for instance, algorithms to build pathloss maps are proposed in [75]. In this paper, the authors propose two kernel-based adaptive algorithms, namely the adaptive projected subgradient method and the multikernel approach with adaptive model selection. Numerical evaluation is done for both a urban scenario and a campus network scenario, using real measurements. The performance of the algorithms is evaluated assuming perfect knowledge of the users' trajectories.

2.1.2.2. Combined channel and mobility context

A joint prediction of channel quality and mobility information is used in [81]. The authors combine information on visited locations and corresponding achieved link quality to provide *connectivity forecast*. A Markov model is used to forecast future channel conditions. Location pre-

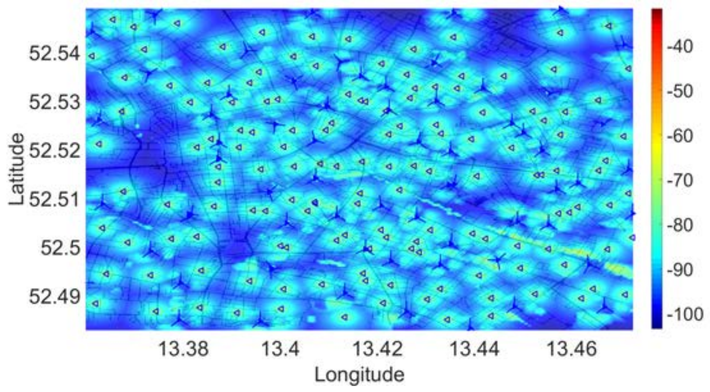


Figure 2.2: Link context example: a pathloss map of Berlin downtown obtained from the data of the MOMENTUM project, where the triangles represent base stations. Pathloss maps are frequently used to predict the evolution of the connection quality in mobile networks.

diction accuracy is approximately 70% for a prediction window of 20 seconds. However, the location information has quite a coarse granularity (of about 100 m). In terms of bandwidth, the proposed model, evaluated on a real dataset, shows an accuracy within 10 KB/s for over 50% of the evaluation period, and within 50 KB/s for over 80% of the time. In [67], the routing metrics in ad hoc wireless networks is assisted by prediction. In particular, the metrics considered in the paper are the average number of retransmissions needed and the time expected to transmit a data packet. The solution anticipates the future signal strength using linear regression on the history of the link quality measurements. Simulations show that the packet delivery ratio is close to 100%, even though it drops to 20% using classical methods.

Prediction is often affected by errors, it is, thus, important to account for their magnitude. This has been considered, for instance, in [59, 76], where the impact of location uncertainties is taken into account. Namely, the authors of [76] show that classical Gaussian Process (GP) wrongly predicts the channel gain in presence of errors, while uncertain GP, which explicitly accounts for location uncertainty, outperforms the former in both learning and predicting the received power. Gains are shown also for a simple proactive resource allocation scenario. Similarly, the second paper [58] discusses a proactive scheduling mechanism that exploits the statistical properties of user demand and channel conditions. Furthermore, the model captures the impact of prediction uncertainties and assesses the optimal gain obtained by the proactive resource scheduler. The authors also propose an asymptotically optimal policy that attains the optimal gain rapidly as the prediction window size increases.

We also considered uncertainties in [6], where a resource allocation algorithm for mobile networks that leverages link quality prediction is proposed. Time series filtering techniques (AutoRegressive Moving Average (ARMA)) are used to predict near term link quality, whereas medium to long term prediction is based on statistical models. We propose a resource allocation optimization framework under imperfect prediction of future available capacity. Simulations are done using a real dataset and show that the proposed solution outperforms the limited horizon optimizer (i.e., when the prediction is done only for the upcoming few seconds) by $10 - 15\%$. Resource allocation is also addressed in [39], which extends the standard PF scheduler of 4G networks to account for data rate prediction obtained through adaptive radio maps.

2.1.2.3. Channel-assisted video optimization

In [68], the authors propose an adaptive mobile video streaming framework, which stores video in the cloud and offers to each user a continuous video streaming adapted to the fluctuations of the link quality. The paper proposes a mechanism to predict the potential available bandwidth in the next time window (of a duration of a few seconds) based on the measurements of the link quality done in the previous time window. A prototype implementation of the proposed framework is used to evaluate the performance. This shows that the prediction has a relative error of about 10% for very short time windows (a couple of seconds) but becomes relatively poor for larger time windows. The video performance is evaluated in terms of “click-to-play” delay, which is halved with the proposed approach.

Video calls are analyzed in [69], where a cross-layer design for proactive congestion control, named Rebera, is proposed. The system measures the real-time available bandwidth and uses a linear adaptive filter to estimate the future capacity. Furthermore, it ensures that the video sending rate never exceeds the predicted values, thereby preventing self-congestion and reducing delays. Performance results with respect to today’s solutions are given for both a testbed and a real cellular network. In [66], the authors propose a hop-by-hop video quality adaptation scheme at the router level to improve the performance of adaptive video streaming in Content Centric Networks (CCNs). In this context, the routers monitor network conditions by estimating the end-to-end bandwidth and proactively decrease the video quality when network congestion occurs. Performance is evaluated considering a realistic large-scale network topology and it is shown that the proposed solution outperforms state of the art schemes in terms of both playback quality and average delay.

A Markov model is used in [70], where information on both channel and buffer states is combined to optimize mobile video streaming. Both an optimal policy as well as a fast heuristic are proposed. A drive test was conducted to evaluate the performance of the proposed solution. In particular, the authors show the proportional dependency between utility and buffer size, as well as the complexity of the two algorithms. Furthermore, a Markov model is adopted to represent different user’s achievable rates [82] and channel states [73]. The transition matrix is derived empirically to minimize the number of video stalls and their duration over a 10-second horizon.

2.1.2.4. Video optimization under uncertainty

The impact of prediction errors is also analyzed for multimedia streaming solutions. In [60], it is proposed a stochastic model of prediction errors based on [4]. Then, an online scheduler that is aware of prediction errors is designed. Namely, based on the expected prediction accuracy, the algorithm determines whether to consider or discard the predicted data rate. A similar model for prediction errors is introduced in [61]. In this case, a Linear Programming (LP) formulation is proposed to trade off spectral efficiency and stalling time. The proposed solution shows good gains with respect to the case without prediction, even when errors occur. LP is used also in [62] to minimize the base station airtime with the constraint of no video interruption. In this case, uncertainties are modeled by using a fuzzy approach. Furthermore, in order to keep track of the previous values of the error, a Kalman filter is used. Simulations are run using synthetic data and show the effect of channel variability on video degradation and average airtime. In [63], the quality of video streaming is increased thanks to bandwidth prediction. Both perfect and uncertain prediction are considered and a robust heuristic is proposed to mitigate the effect of prediction errors when adapting the video bitrate. In [64, 65], a predictive resource allocation robust to rate uncertainties is proposed. The authors propose a framework that provides quality guarantees with the objective of minimizing energy consumption. Both optimal gradient-based and real-time guided heuristic solutions are presented. In [64] both Gaussian and Bernstein approximation are used to model rate uncertainties, whereas [65] considers only the former one. Similarly, [164] provides predictive Quality-of-Service (QoS) over wireless ATM networks: given the TDMA nature of these networks, these schemes optimize the number of allocated time slots depending on the characteristics of the stream and the link.

2.1.2.5. Efficiency bounds and approximations for multimedia streaming applications

Other papers ([7,8,52–57]) analyze resource allocation optimization assuming that the future channel state is perfectly known. Differing on the final objectives, these papers adopt similar methods: they first devise a problem formulation from which an optimal solution can be obtained (using standard optimization techniques), then they propose sub-optimal approaches and on-line algorithms to obtain an approximation of the optimal solution. Furthermore, all these papers leverage a buffer to counteract the randomness of the channel. For instance, in case a given amount of information has to be gathered within a deadline, the buffer allows the system to optimize (for a given objective function) the resource allocation while meeting the deadline.

Concerning their goals, energy-efficiency is the primary objective in [52, 53], which is optimized by allowing the network base stations to be switched off once the users' streaming requirements have been satisfied. Simulations show that an energy saving up to 80% with respect to the baseline approach can be achieved and that the performance of the heuristic solution is quite close to the optimal (but impractical) Mixed-Integer Linear Programming (MILP) approach. Buffer size is investigated in [56], where the author introduces a linear formulation that minimizes the amount for resources assigned to non-real time video streaming with constraints on the user's playout buffer. Results are shown for a scenario with both video and best effort users and highlight the gain in terms of required resources to serve the video users as well as data rate for the best effort users.

The trade-off between streaming interruption time and average quality is investigated in [54, 55] by devising a mixed-integer quadratically constrained problem which computes the optimal download time and quality for video segments. Then, the authors propose a set of heuristics tailored to greedily optimize segment scheduling according to a specific objective function, e.g., maximum quality, minimum streaming interruption, or fairness. Similar objectives are tackled in [7,8] in a lexicographic approach, so that streaming continuity is always prioritized over quality. They first propose a heuristic for the lateness-quality problem that performs almost as good as the MILP formulation. Then, they extend the MILP formulation to include QoS guarantees and they introduce an iterative approximation based on a simpler LP formulation. A further heuristic approach is devised in [57] and accounts for the buffer and channel state prediction. The proposed approach maximizes the streaming quality while guaranteeing that there are no interruptions.

2.1.2.6. Cognitive radio maps

CRs are context-aware wireless devices that adapt their functionalities to changes in the environment. They have been recently used [165–167] to obtained the so-called Radio Environment Map (REM): a multi-dimensional database containing a wide set of information ranging from regulations to spectrum usage.

For instance, REM are used to predict spectrum availability in CR [165]: the paper exploits cognitive maps to provide contextual information for predictive machine learning approaches such as Hidden Markov Models (HMM), ANN and regression techniques. The construction of these maps is discussed in [166] and the references therein, while their use as enabler for CR networks is analyzed in [167].

In the context of anticipatory networking, REMs are often used as a source of contextual information for the actual prediction technique adopted, rather than as prediction tools themselves. [168, 169] present two surveys of methodologies and measurement campaigns of spectrum occupancy. In particular, [168] proposes a cautionary approach to account for measurement uncertainty, while [169] exploits predictors to provide the future channel status. In addition, prediction through machine learning approaches is addressed in [170], where different techniques are compared to assess future channel availability.

Imperfect measurements are dealt with in [171] which models the problem as a repeated game and maximize the total network payoff. However, in cognitive networks, the channel status depends on the activity of primary users: [172] surveys the models proposed so far to describe this and that can be used to drive prediction in this area. Once the activity of primary users is available or predicted, it is possible to control the activity of secondary users in order to guarantee the agreed QoS to the formers [173, 174]: these papers compute the feasible cognitive interference region in order to allow secondary users' communication respecting primary users' rights. The utilization of spectrum opportunity describes the probability of a secondary user to exploit a free communication slot and is described in [175].

A similar form of opportunistic spectrum usage goes under the name of white spaces [176]: i.e. channels that are unused at specific location and time. CRs can take advantage of these frequencies thanks to dynamic spectrum access. Finally, [177] describes how to exploit CR to realize a complete smart grid scenario; [178] describes how to exploit channel bonding to increase the bandwidth and decrease the delay of CR.

2.1.3. Traffic Context

Although related to the previous context, the papers discussed in this section leverage information collected from higher layers of the protocol stack. For instance, solutions falling in this category try to predict, among other parameters, the number of active users in the network and the amount of traffic they are going to produce. Similarly, but from the perspective of a single user, the prediction can target the data rate that a streaming application is going to achieve in the near term.

2.1.3.1. Traffic analysis and characterization

Analyzing mobile traffic is fundamental for long-term network optimization and reconfiguration. To this end, several pieces of work have addressed such research topics recently.

The work in [109] build regressors for different performance metrics at multiple spatio-temporal granularity for mobile cellular networks. In particular, the authors focus on the characterization of per-device throughput, base station throughput and device mobility. A one-week nation-wide cellular network dataset is collected through proprietary traffic inspection tools placed in the operator network and are used to characterize the per-user traffic, cell-aggregate traffic and to perform further spatio-temporal correlation analysis.

A similar scope is addressed by [112] which, on the other hand, focuses more on core network measurements. Flow level mobile device traffic data are collected from a cellular operator's core network and are used to characterize the IP traffic patterns of mobile cellular devices. More recently, the authors of [110] studied traffic prediction in cloud analytics and prove that optimizing the choice of metrics and parameters can lead to accurate prediction even under high latency.

2.1.3.2. Traffic prediction

The prediction of several traffic performance parameters can be used in many applications. For instance, a predictive framework that anticipates the arrival of upcoming requests is used in [88] to prefetch the needed content at the mobile terminal. The authors propose a theoretical framework to assess how the outage probability scales with the prediction horizon. The theoretical framework accounts for prediction errors and multicast delivery. Along the same line, queue modeling [86] and analysis [84] is used to predict the upcoming workloads in a lookahead time window. Leveraging the workload prediction, a multi-slot joint power control and scheduling problem is formulated to find the optimal assignment that minimizes the total cost [86] or maximizes the QoS [84].

Multimedia optimization is the focus in [113]. By predicting throughput, packet loss and transmission delay half a second in advance, the authors propose to dynamically adjust application-level parameters of the reference video streaming or video conferencing services including the compression ratio of the video codec, the forward error correction code rate and the size of the de-jittering buffer. The authors of [108] propose to use a database of events (concerts, gatherings, etc.) to improve the quality of the traffic prediction in case of unexpected traffic patterns and in [99], where a general predictive control framework along with Kalman filter is proposed to counteract the impact of network delay and packet loss. The objective of [111] is to build a model for user engagement as a function of performance metrics in the context of video streaming services. The authors use a supervised learning approach based on average bitrate, join time, buffering ratio and buffering to estimate the user engagement. Finally, inter-download time can be modeled [117] and subsequently predicted for quality optimization.

Energy-efficient resource scheduling in mobile radio networks is addressed in [92]. The paper introduces a Mixed Non-Linear Program (MNLP) which returns on a slot basis the optimal allocation of resources to users and the optimal users-cell association pattern. The proposed model leverages optimal traffic predictors to obtain the expected traffic conditions in the following slots. Radio resource allocation in mobile radio networks is addressed also in [95] and later by the same authors in [94]; the target is to design a predictive framework to optimally orchestrate the resource allocation and network selection in case one operator owns multiple access networks. The predictive framework aims at minimizing the expected time average power consumption while keeping the network (user queues) stable. The core contribution of [96, 97] is the use of deep learning techniques to predict the upcoming video traffic sessions; the prediction outcome is then used to proactively allocate the resources of video servers to these future traffic demands.

2.1.3.3. Throughput prediction

This section considers works predicting or using prediction based on the expected throughput, rather than the traffic prediction. A common characteristic of the work described here is that the spatio-temporal correlation is exploited in the prediction phase of the expected throughput.

Quite a few early works studied how to effectively predict the obtainable data rate. In particular, long term prediction [101] with 12-hour granularity allows to estimate aggregate demands up to 6 months in advance. Shorter and variable time scales are studied in [103, 104] adopting ARIMA and Generalized AutoRegressive Conditionally Heteroskedastic (GARCH) techniques.

In [85], the authors propose a dynamic framework to allocate downlink radio resources across multiple cells of 4G systems. The proposed framework leverages context information of three types: radio maps, user's location and mobility, as well as application-related information. The authors assume that a forecast of this information is available and can be used to optimize the resource allocation in the network. The performance of the proposed solution is evaluated through simulation for the specific use case of video streaming. Geo-localized radio maps are also exploited in [89]. Here the optimization is performed at the application layer by letting adaptive video streaming clients and servers dynamically change the streaming rate on the basis of the current bandwidth prediction from the bandwidth maps. The empirical collection of geo-localized data rate measures is also addressed in [102] which introduces a dataset of adaptive Hypertext Transfer Protocol (HTTP) sessions performed by mobile users.

The anticipation of the upcoming throughput values is often applied to the optimization of adaptive video streaming services. In this context, Yin *et al.* [90] leverage throughput prediction to optimally adapt the bit rate of video encoders; here, prediction is based on the harmonic mean of the last k throughput samples.

The work in [115] considers the problem of predicting end-to-end quality of multi-hop paths in community WiFi networks. The end-to-end quality is measured by a linear combination of the expected transmission count across all the links composing the multi-hop path. The authors resort to a real data set of a WiFi community network and test several predictors for the end-to-end quality.

The work in [91] resorts to a model-based throughput predictor in which the throughput of a Dynamic Adaptive Streaming over HTTP (DASH)-based video streaming service is assumed to be a random variable with Beta-like distribution whose parameters are empirically estimated within an observation time window. Building on this estimate, the authors propose a MNLP with a concave objective function and linear constraints. The program is implemented as a multiple choice knapsack problem and solved using commercial solvers. Along the same lines, the optimization of a DASH-based video streaming service is addressed in [93], where the authors propose an adaptive video streaming framework based on a smoothed rate estimate for the video sessions.

In [106, 114] the authors build on the conjecture that video sessions sharing the same critical features have similar QoE (e.g., re-buffering, startup latency, etc.). Consequently, first clustering techniques are applied to group similar video sessions, and then throughput predictors based on HMMs are applied to each cluster to dynamically adapt the bit rate of the video encoder to the predicted throughput samples. The work in [98] considers the scenario where a small cell is used to deliver video content to a highly dense set of users. The video delivery can also be supported in a distributed way by end-user devices storing content locally. A control-theoretic framework is proposed to dynamically set the video quality of the downloaded content while enforcing stability of the system.

2.1.4. Social Context

The work on anticipatory networking leveraging social context exploits *ex ante* or *ex post* information on social-type relationships between agents in the networking environment. Such information may include: the network of social ties and connections, the user's preference on contents, measures on user's centrality in a social network, and measures on users' mobility habits. The aforementioned context information is leveraged in three main application scenarios: caching at the edge of mobile networks, mobility prediction, and downlink resource allocation in mobile networks.

2.1.4.1. Social-assisted caching

To limit the load in 5G backhaul networks, the authors of [118, 128, 129] propose two schemes to proactively move contents closer to the end users. In [118], caching happens at the small cells, whereas in [128, 129] contents can be proactively downloaded by a subset of end users which then re-distribute them via device-to-device (D2D) communication. The authors first define two optimization problems which target the load reduction in the backhaul (caching at small cells) and in the small cell (caching at end users), respectively, then heuristic algorithms based on machine learning tools are proposed to obtain sub-optimal solutions in reasonable processing time. The heuristic first collects users' content rating/preferences to predict the popularity matrix $\mathbf{P}_m$. Then, content is placed at each small cell in a greedy way starting from the most popular ones until a storage budget is hit. The first algorithmic step of caching at the end users is to identify the K most connected users and to cluster the remaining ones in communities. Then it is possible to characterize the content preference distributions within each community and greedily place contents at the cluster heads. In [129], the prediction leverages additional information on the underlying structure of content popularity within the communities of users.

In [123, 124], it is argued that proactive caching of delay intolerant content based on user preferences is subject to prediction uncertainties that affect the performance of any caching scheme. In [123], these uncertainties are modeled as probability distributions of content requests over a given time period. The authors provide lower bounds on the content delivery cost given that the probability distribution for the requests is available. They also derive caching policies that achieve this lower bound asymptotically. It is shown that under uniform uncertainty, the proposed policy breaks down to equally spreading the amount of predicted content data over the horizon of the prediction window. Another approach to solve the same problem is used in [124], where personalized content pricing schemes are deployed by the service provider based on user preferences in order to enhance the certainty about future demand. The authors model the pricing problem as an optimization problem. Due to the non-convex nature of their model, they use an iterative sub-optimal solution that separates price allocation and proactive download decisions.

Joint mobility and popularity prediction for content caching at small cell base stations is studied in [121]. Here, the authors propose a heuristic caching scheme that determines whether a particular content item should be cached at a particular base station by jointly predicting the mobility pattern of users that request that item as well as its popularity, where popularity prediction is performed using the inter-arrival times of consecutive requests for that object. They conclude that the joint scheme outperforms caching with only mobility and only popularity models. A similar problem is addressed in [138]: the authors consider a distributed network of femto base stations, which can be leveraged to cache videos. The authors study where to cache videos such that the average sum delay across all the end users is minimized for a given video content popularity distribution, a given storage capacity and an arbitrary model for the wireless link. A greedy heuristic is then proposed to reduce the computational complexity.

2.1.4.2. Social-assisted matching game theory

Matching game theory [135] can be used to allocate networks resources between users and base stations, when social attributes are used to profile users. For instance, users and base stations can rank one another to capture users' similarities in terms of interests, activities and interactions. Thus, it is possible to create social utility functions that can be further used to control a distributed matching game. In [80], a self-organizing, context-aware framework for D2D resource allocation is proposed that exploits the likelihood of strongly connected users to request similar contents. The solution is shown to be computationally feasible and to offer substantial benefits when users' social similarities are present. A similar approach is used in [116] to deal with joint millimeter and micro wave dual base station resource allocation, in [41] for user base station association in small cell networks, and in [139] to optimize D2D offloading techniques. Caching in small cell networks can also be addressed as a many-to-many matching game [134]: by matching video popularity among users most frequently served by a given server it is possible to devise caching policies that minimize end-users' delays. Simulations show the approach is effective in small cell networks.

2.1.4.3. Social-assisted mobility prediction

To reduce the resource wasted during active scanning in IEEE 802.11 networks, the authors of [125] propose a mobility prediction tool to anticipate the next access point a WiFi user is moving to. The proposed solution is based on context information on the handoffs which were performed in the past; specifically, the system stores centrally a time varying handoff table which is then fed into an ARIMA predictor which returns the likelihood of a given user to handoff to a specific access point. The quality of the predictor is measured in terms of signaling reduction due to active scanning.

User mobility prediction is also exploited in [132]. The authors leverage information coming from the social platform Foursquare to predict user mobility on coarse granularity. The *next check-in problem* is designed to forecast the next place in an urban environment which will be most likely visited by a user. The authors build a time-stamped dataset of "check-ins" performed by Foursquare users over a period of one month across several venues worldwide. A set of features is then defined to represent user mobility including user mobility features (e.g., number of historical visits to specific venues or categories of venues, number of historical visits that friends have done to specific venues), global mobility features (e.g., popularity of venues, distance between venues, transition frequency between couples of venues), and temporal features which measures the historical check-ins over specific time periods. A supervised classification problem is then trained with the obtained features to predict the next check-in venue. Linear regression and M5 decision trees are the chosen classifiers.

Along the same lines, the mobility of users in urban environments is characterized in [137]. Different from the previous work which only exploits social information, the authors also leverage physical information about the current position of moving users. A probabilistic model of the mobile users' behavior is built and trained on a real life dataset of user mobility traces. A social-assisted mobility prediction model is proposed in [136], where a variable-order Markov model is developed and trained on both temporal features (i.e., when users were at specific locations) and social ones (i.e., when friends of specific users were at a given location). The accuracy of the proposed model is cross-validated on two user-mobility datasets.

2.1.4.4. Social-assisted radio resource allocation

Elastic traffic optimization in the downlink of mobile radio networks is the main objective of [87, 119]. The main idea is "enriching" the downlink scheduler with contextual information to make better decisions in the allocation of the radio resources. Besides classical network-side context including the cell load and the current channel quality indicator which are widely used in the literature to steer the scheduling, the authors propose to include user-side features which generically capture the satisfaction degree of the user for the reference application. Namely, the authors introduce the concept of a *transaction*, which represents the atomic data download requested by the end user (e.g., a web page download via HTTP, an object download via HTTP or a file download via File Transfer Protocol (FTP)). For each transaction and for each application, a utility function is defined capturing the user's sensitivity with respect to the transmission delay and the expected completion time. The functional form of this utility function depends on the type of application which "generated" the transaction; as an example, the authors make the distinction between transactions from applications which are running in the foreground and the background on the user's terminal. For the sake of presentation, a parametric logistic function is used to represent the aforementioned utility. The authors then formulate an optimization problem to maximize the sum utility across all the users and transactions in a given mobile radio cell and design a greedy heuristic to obtain a sub-optimal solution in reasonable computing time. The proposed algorithm is validated against state-of-the-art scheduling solutions (PF / weighted PF scheduling) through simulation on synthetic data mimicking realistic user distributions, mobility patterns and traffic patterns.

Social-oriented techniques related to the popularity of the end users are leveraged also in [120] where the authors target the performance optimization of downlink resource allocation in future generation networks. The utility maximization problem is formulated with the utility being a combination (product) of a network-oriented term (available bandwidth) and a social-oriented term (social distance). The social-oriented term is defined to be the degree centrality measure [148] for a specific user. The proposed problem is sub-optimally solved through a heuristic which is finally validated using synthetic data.

Spatial traffic of base stations in a cellular network is predicted in [133] by applying the idea of social networks to base stations. Here, the base stations themselves create a social network and a social graph is created between them based on the spatial correlation of the traffic of each of them. The correlation is calculated using the Pearson coefficient. Based on the topology of the social graph, the most important base stations are identified and used for traffic prediction of the entire network, which is done using SVM. The authors conclude that with the traffic data of less than 10% of the base stations, effective prediction with less than 20% mean error can be achieved.

Table 2.2: Context Classification Summary: each context is associated to its most popular applications, prediction techniques, optimization methods and main notable characteristics.

Context	Applications	Prediction[a]	Optimization	Remarks
Geographic [11-26, 28, 29, 31-35, 37, 38, 41-46, 131]	Mobility prediction Multimedia streaming Broadcast Resource allocation Duty cycling	1st Probabilistic 2nd Regression 3rd Time series 4th Classification	1) Prediction to define convex optimization problems 2) Prediction as the optimization objective	1) Prediction accuracy is inversely proportional to the time scale and granularity 2) High prediction accuracy if periodicity and/or trends are present 3) Prediction is more effectively used in delay tolerant applications
Link [30, 47-70, 72-79, 129, 158]	Channel forecast Resource allocation Network mapping Routing Multimedia streaming	1st Regression 2nd Time series 3rd Probabilistic 4th Classification	1) Markov decision process is used when statistical knowledge is available 2) Convex optimization is preferred when it is possible to perform accurate forecast	1) Channel quality maps can be effectively used to improve networking 2) Movement dynamics affect the prediction effectiveness 3) Channel is most often predicted by means of functional regression or Markovian models
Traffic [92-102 104-120 138 145 156 165]	Traffic analysis Resource allocation Multimedia streaming	1st Regression 2nd Classification 3rd Probabilistic	1) Maps are used to deterministically drive the optimization 2) Convex optimization problems can be formulated to obtain bounds	1) Improved long-term network optimization and reconfiguration 2) Traffic distribution is skewed both with regards of users and locations 3) Traffic has a strong time periodicity 4) Geo-localized information can be used
Social [40 121-140 148 149 154 157 159]	Network caching Mobility prediction Resource allocation Multimedia streaming	1st Classification 2nd Regression 3rd Time series 4th Probabilistic	1) Formal optimization problems are usually impractical 2) Game theoretic and heuristics are more efficient	1) A fraction of social information can be accurately predicted 2) Prediction obtained from social information is usually coarse 3) Social information prediction can effectively improve application performance

[a] Ranking based on the number of papers reviewed in this survey using the predictor.

2.1.5. Summary

We conclude this section by summarizing the main takeaways in terms of application and objective for which different context types can be used. Table 2.2 provides a synthesis of the main considerations: each context is associated with its typical applications, prediction methodologies (ordered by decreasing popularity), optimization approaches and general remarks.

Mobility prediction – The predictability of user mobility can be potentially very high (93% potential predictability in user mobility as stated in [42]), despite the significant differences in the travel patterns. As a matter of fact, many studies forecast users' mobility by means of a variety of techniques. For predicting trajectories, characterized by sequences of discretized locations indicated by cell identitys (IDs) or road segments, fixed-order Markov models or variable-order Markov models are the most promising tools, while for continuous trajectories, regression techniques are widely used. To enhance the prediction accuracy, the most popular ones leverage geographic information: GPS data, cell records and received signal strength are used to obtain precise and frequent data sampling to locate users on a map. However, the movements of an individual are largely influenced by those of other individuals via social relations. Several papers analyze social information and location check-ins to find recurrent patterns. In this second case the dataset is usually sparser, which may limit the accuracy of the prediction.

Network efficiency – The most frequent objective in anticipatory networking is predicting and optimizing the network efficiency (i.e., increasing the performance of the network while using the same amount of resources). We found papers exploiting all four types of context to achieve this. As such, objectives and constraints cover the whole attribute space. Improving network efficiency is likely to become the main driver for including anticipatory networking solutions in next generation networks.

Multimedia streaming – Multimedia streaming and, in particular, video on demand are the main source of data traffic in 4G networks. Therefore, 5G networks are expected to continue and even increase this trend. As a consequence, several anticipatory networking solutions focus on the optimization of this service. All the context types have been used to this extent and each has a different merit: social information is needed to predict when a given user is going to request a given content, combined geographic and social information allows the network to cache that content closer to where it will be required and physical channel information can be used to optimize the resource assignment.

Network offloading – Mobility prediction can be used to handover communications between different technologies to decrease network congestion, improve user experience, reduce users' costs and increase energy efficiency.

Cognitive networking – Physical channel prediction can be exploited for cognitive networking and for network mapping. The former application allows secondary users to access a shared medium when primary subscribers left resource unused, thus, predicting when this is going to happen will highly improve the effectiveness of the solution. The latter, instead, exploits link information to build networking maps.

Throughput- and traffic-based applications – Traffic information is usually studied to be, first, modeled and, subsequently, predicted. Traffic models and predictors are then used to improve networking efficiency by means of resource allocation, traffic shaping and planning.

2.2. Prediction Methodologies for Anticipatory Networking

This section summarizes some selected prediction methods for the types of context introduced above. The methods belongs into four main categories: *time series methods*, *similarity-based classification*, *regression analysis*, and *statistical methods for probabilistic modeling*. The goal of the prediction handbook is to show *which methods work in which situation.* In fact, selecting the appropriate prediction method requires to analyze the prediction variables and the model constraints with respect to the application scenario. This section concludes with a series of takeaways that summarize some general principles for selection of prediction methods based on the scenario analysis.

2.2.1. Time Series Predictive Modeling

A time series is a set of time-stamped data entries which allows a natural association of data collected on a regular or irregular time basis. Time series are widely used to store wireless networks contents and which frequently reveal some temporal correlation. For example, the trajectory of the mobile device can be characterized by successive time-stamped locations obtained from geographical measurements; individual social behavior can be expressed through time-evolving events; traffic loads modeled in time series can be leveraged for network planning and controlling. Fig. 2.3(a) and 2.3(b) illustrate two time series of per-cell and per-city aggregated uplink and downlink data traffic, where temporal correlation is clearly recognizable.

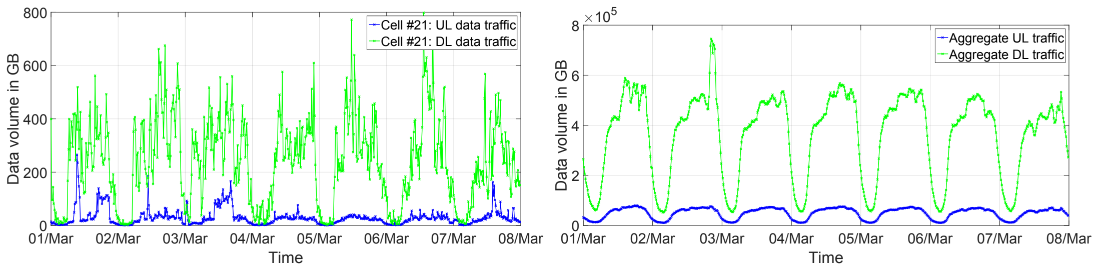
The two most widely used time series models are based on linear dynamic systems: 1) AutoRegressive Moving Average (ARMA), and 2) Kalman filters. In what follows we introduce the two techniques together with examples of context prediction in wireless networks and their extensions to nonlinear systems.

2.2.1.1. Autoregressive and moving average models

Consider a univariate time series $\{X_t : t \in \mathcal{T}\}$, where $\mathcal{T}$ denotes the set of time indices. The general ARMA model, denoted by $\mathrm{ARMA}(p, q)$, has p AR terms and q Moving Average (MA) terms, given by

$$X_t = Z_t + \sum_{i=1}^{p} \phi_i X_{t-i} + \sum_{j=1}^{q} \theta_j Z_{t-j} \tag{2.1}$$

where Z_t is the process of the white noise errors, and $\{\phi_i\}_{i=1}^{p}$ and $\{\theta_j\}_{j=1}^{q}$ are the parameters. The ARMA model is a generalization of the simpler AR and MA models that can be obtained for



(a) Uplink and downlink traffic. (b) Aggregated traffic.

Figure 2.3: Example of time series: Traffic load (aggregated every 15 minutes) for a week in March 2015 in Rome, Italy. Data source from Telecom Italia's Big Data Challenge.

$q = 0$ and $p = 0$ respectively. Using the *lag operator* $L^i X_t := X_{t-i}$ the model becomes

$$\phi(L)X_t = \theta(L)Z_t \tag{2.2}$$

where $\phi(L) := 1 - \sum_{i=1}^{p} \phi_i L^i$ and $\theta(L) := 1 + \sum_{j=1}^{q} \theta_j L^j$.

The fitting procedure of such processes assumes *stationarity*. However, this property is seldom verified in practice and *non-stationary* time series need to be stationarized through differencing and logging. The ARIMA model generalizes ARMA models for the case of non-stationary time series: a non seasonal ARIMA model $\mathrm{ARIMA}(p, d, q)$ after d differentiations reduces to an $\mathrm{ARMA}(p, q)$ of the form

$$\phi(L)\Delta^d X_t = \theta(L)Z_t, \tag{2.3}$$

where $\Delta^d = (1 - L)^d$ denotes the dth difference operator.

The prediction of traffic load in wireless or IP backbone networks using autoregressive models has been the focus of numerous studies. The stationarity analysis often provides important clues for selecting the appropriate model. For instance, in [101] a low-order ARIMA model is applied to capture the non-stationary short memory process of traffic load, while in [103] a Gegenbauer ARMA model is used to specify long memory processes under the assumption of stationarity. Similar models are applied to mobility- or channel-related contexts. In [125], an exponential weighted moving average, equivalent to $\mathrm{ARIMA}(0, 1, 1)$, is used to forecast handoffs. In [27,79], AR models are applied to predict future signal-to-noise ratio values and user positions, respectively. If the variance of the data varies with time, as in [104] for data traffic, and can be expressed using an ARMA, then the whole model is referred to as GARCH.

2.2.1.2. Kalman filter

The time series analysis of linear dynamic systems is often performed using Kalman filters, which track the estimated system state and its uncertainty variance. In the anticipatory networking literature, Kalman filters have been mainly adopted to model the linear dependence of the system states based on historical data.

Consider a multivariate time series $\{\mathbf{x}_t \in \mathbb{R}^n : t \in \mathcal{T}\}$, the Kalman filter addresses the problem of estimating state $\mathbf{x}_t$ that is governed by the linear stochastic difference equation

$$\mathbf{x}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t + \mathbf{w}_t,\ t = 0, 1, \dots, \tag{2.4}$$

where $\mathbf{A}_t \in \mathbb{R}^{n \times n}$ expresses the state transition, and $\mathbf{B}_t \in \mathbb{R}^{n \times l}$ relates the optional control input $\mathbf{u}_t \in \mathbb{R}^l$ to the state $\mathbf{x}_t \in \mathbb{R}^n$. The random variable $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_t)$ represents a multivariate normal noise process with covariance matrix $\mathbf{Q}_t \in \mathbb{R}^{n \times n}$. The observation $\mathbf{z}_t \in \mathbb{R}^m$ of the true state $\mathbf{x}_t$ is given by

$$\mathbf{z}_t = \mathbf{H}_t \mathbf{x}_t + \mathbf{v}_t, \tag{2.5}$$

where $\mathbf{H}_t \in \mathbb{R}^{m \times n}$ maps the true state space into the observed space. The random variable $\mathbf{v}_t$ is the observation noise process following $\mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_t)$ with covariance $\mathbf{R}_t \in \mathbb{R}^{n \times n}$. Kalman filters iterate between 1) predicting the system state with Eq. (2.4) and 2) updating the model according to Eq. (2.5) to refine the previous prediction. The interested reader is referred to [149] for more details.

In [30, 179], Kalman filters are used to study users' mobility. Wireless channel gains are studied in [74] with KKF, while the authors of [100] adopt the technique to predict short-term traffic volume. The extended Kalman filter adapts the standard model to nonlinear systems via online Taylor expansion. According to [180], this improves shadow/fading estimation.

2.2.2. Similarity-based Classification

Similarity-based classification involves finding the inherent structures within datasets. The main idea is that similarity patterns in a dataset can be used to predict unknown data or missing features. A typical application of these concept is recommendation systems where users give a score to items and the system tries to infer similarities among users and scores to predict the missing entries.

These techniques are unsupervised learning methods, since categories are not predetermined, but are inferred from the data. They are applied to datasets exhibiting one or more of the following properties: 1) entries of the dataset have many attributes, 2) no law is known to link the different features, and 3) no classification is available to manually label the dataset.

In what follows, we briefly review the similarity-based classification tools that have been used in the anticipatory networking literature accounted for in this survey.

2.2.2.1. Collaborative filtering

Recommendation systems usually adopt Collaborative Filtering (CF) to predict unknown opinions according to user's and/or content's similarities. While a thorough survey is available in [150], here, we just introduce the main concepts related to anticipatory networking.

CF predicts the missing entries of a $n_c \times n_u$ matrix $\mathbf{Y} \in \mathcal{A}^{n_c \times n_u}$, mapping n_c users to n_u

contents through their opinions which are taken from an alphabet $\mathcal{A}$ of possible ratings. Thus, the entry $y_{ik}, i \in \{1, \dots, n_c\}, k \in \{1, \dots, n_u\}$ expresses how much user k likes content i. An auxiliary matrix $\mathbf{R} \in [0,1]^{n_c \times n_u}$ expresses whether user k evaluated content i ($r_{ik} = 1$) or not ($r_{ik} = 0$).

To predict the missing entries of $\mathbf{Y}$ the feature learning approach exploits a set of n_f features to represent contents' and users' similarities and defines two matrices $\mathbf{X} \in [0,1]^{n_c \times n_f}$ and $\mathbf{\Theta} \in \mathcal{A}^{n_u \times n_f}$, whose entries x_{ij} and θ_{kj} represent how much content i is represented by feature j and how high user k would rate a content completely defined by feature j, respectively. The new matrices aim to map $\mathbf{Y}$ in the feature space and they can be computed by:

$$\underset{\mathbf{X},\mathbf{\Theta}}{\operatorname{argmin}} \sum_{i,k:r_{ik}=1} (\mathbf{x}_{i*} \boldsymbol{\theta}_{k*}^T - y_{ik})^2, \tag{2.6}$$

where $\mathbf{x}_{i*} := (\mathrm{col}_i \mathbf{X}^T)^T$ denotes the i-th row of matrix $\mathbf{X}$. Note that in (2.6) the regularization terms are omitted. Solving (2.6) amounts to obtain a matrix $\tilde{\mathbf{Y}} = \mathbf{X}\mathbf{\Theta}^T$ which best approximates $\mathbf{Y}$ according to the available information ($i, k : r_{ik} = 1$). Finally, $\tilde{y}_{ik} = \mathbf{x}_{i*}\boldsymbol{\theta}_{k*}^T$ predicts how user k with parameters $\boldsymbol{\theta}_{k*}$ rates content i having feature vector $\mathbf{x}_{i*}$.

Other applications of CF are, for instance, network caching optimization [127, 130], where communication efficiency is optimized by storing contents where and when they are predicted to be consumed. Similarly, location-based services [132] predict where and what to serve to a given user.

2.2.2.2. Clustering

When the classification objective is finding groups of similar items within data sets, the adopted method is called clustering. The following provides an introduction to K-means, which is among the most commonly-used clustering techniques in anticipatory networking. The interested reader is referred to [151] for a complete review.

K-means splits a given dataset into K groups without any prior information about the group structure. The basic idea is to associate each observation point from a dataset $\mathcal{X} := \{\mathbf{x}_i \in \mathbb{R}^n : i = 1, \dots, M\}$, to one of the centroids in set $\mathcal{M} := \{\boldsymbol{\mu}_j \in \mathbb{R}^n : j = 1, \dots, K\}$. The centroids are optimized by minimizing the intra-cluster sum of squares (sum of distance of each point in the cluster to the K centroids), given by

$$\underset{\mathcal{C},\mathcal{M}}{\operatorname{minimize}} \sum_{j=1}^{K} \sum_{i=1}^{M} c_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2, \tag{2.7}$$

where $\mathcal{C} := \{c_{ij} \in \{0,1\} : i = 1, \dots, M, j = 1, \dots, K\}$ associates entry $\mathbf{x}_i$ to centroid $\boldsymbol{\mu}_j$. No entry can be associated to multiple centroids ($\sum_{j=1}^{K} c_{ij} = 1, \forall i \in \mathcal{M}$).

A few examples from the literature includes [78], where clustering is applied in anticipatory networking to build a data-driven link model, [35] where it is used to find similarities within

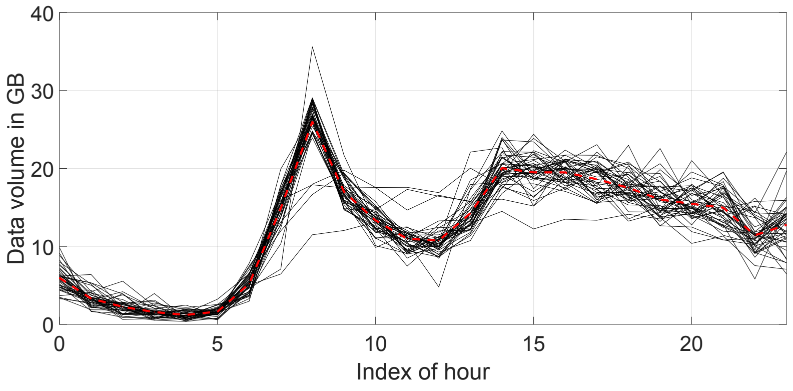


Figure 2.4: Example of a functional dataset: WiFi traffic in Rome depending on hour of the day. Data source from Telecom Italia's Big Data Challenge.

vehicular paths, [108] which identifies social events that might impact network performance, and [112] for device types.

2.2.2.3. Decision Trees

A supervised version of clustering is *decision tree learning* (the interested reader is referred to [152] for a survey on the topic). Assuming that each input observation is mapped to a consequence on its target value (such as reward, utility, cost, etc.), the goal of decision tree learning is to build a set of rules to map the observations to their target values. Each decision branches the tree into different paths that lead to leaves representing the class labels. With prior knowledge, decision trees can be exploited for location-based services [132], for identifying trajectory similarities [36], and for predicting the QoE for multimedia streams [111]. For continuous target variables, regression trees can be used to learn trends in network performance [113].

2.2.3. Regression Analysis

When the interest lies in understanding the relationship between different variables, regression analysis is used to predict dependent variables from a number of independent variables by means of so-called regression functions. In the following, we introduce three regression techniques, which are able to capture complex nonlinear relationships, namely *functional regression*, *support vector machines* and *artificial neural networks*.

2.2.3.1. Functional regression

Functional data often arise from measurements, where each point is expressed as a function over a physical continuum (e.g., Fig. 2.4 illustrates the example of aggregated WiFi traffic as a function of the hour of the day). Functional regression has two interesting properties: smoothness allows to study derivatives, which may reveal important aspects of the processes generating the data, and the mapping between original data and the functional space may reduce the dimensionality of the problem and, as a consequence, the computational complexity [154]. The commonly encountered form of function prediction regression model (scalar-on-function) is given by [153]:

$$Y_i = B_0 + \int X_i(z)B(z)dz + E_i \tag{2.8}$$

where $Y_i, i = 1, \dots, M$ is a continuous response, $X_i(z)$ is a functional predictor over the variable z, $B(z)$ is the functional coefficient, B_0 is the intercept, and E_i is the residual error.

Functional regression methods are applied in [110] to predict traffic-related LTE metrics (e.g., throughput, modulation and coding scheme, and used resources) showing that cloud analytics of short-term LTE metrics is feasible. In [131], functional regression is used to study churn rate of mobile subscribers to maximize the carrier profitability.

2.2.3.2. Support vector machines

SVM is a supervised learning technique that constructs a hyperplane or set of hyperplanes (linear or nonlinear) in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. In this survey we introduce the SVM for classification, and the same principle is used by SVM for regression. Consider a training dataset $\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^n, y_i \in \{-1, 1\}, i = 1, \dots, M\}$, where $\mathbf{x}_i$ is the i-th training vector and y_i the label of its class. First, let us assume that the data is linearly separable and define the linear separating hyperplane as $\mathbf{w} \cdot \mathbf{x} - b = 0$, where $\mathbf{w} \cdot \mathbf{x}$ is the Euclidean inner product. The optimal hyperplane is the one that maximizes the *margin* (i.e., distance from the hyperplane to the instances closest to it on either side), which can be found by solving the following optimization problem:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to} && y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \; \forall i \in \{1, \dots, M\}. \end{aligned} \tag{2.9}$$

Fig. 2.5(a) shows an example of linear SVM classifier separating two classes in $\mathbb{R}^2$.

If the data is not linearly separable, the training points are projected to a high-dimensional space $\mathcal{H}$ through a nonlinear transformation $\phi : R^n \to \mathcal{H}$. Then, a linear model in the new space is built, which corresponds to a nonlinear model in the original space. Since the solution of (2.9) consists of inner products of training data $\mathbf{x}_i \cdot \mathbf{x}_j$, for all i, j, in the new space the solution is in the form of $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$. The *kernel trick* is applied to replace the inner product of basis functions by a *kernel function* $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$ between instances in the original input space, without explicitly building the transformation ϕ.

The Gaussian kernel $K(\mathbf{x}, \mathbf{y}) := \exp(\gamma||\mathbf{x} - \mathbf{y}||^2)$ is one of the most widely used kernels in the literature. For example, it is used in [33] to predict user mobility. In [75], the authors propose an algorithm for reconstructing coverage maps from path-loss measurements using a kernel method. Nevertheless, choosing an appropriate kernel for a given prediction task remains one of the main challenges.

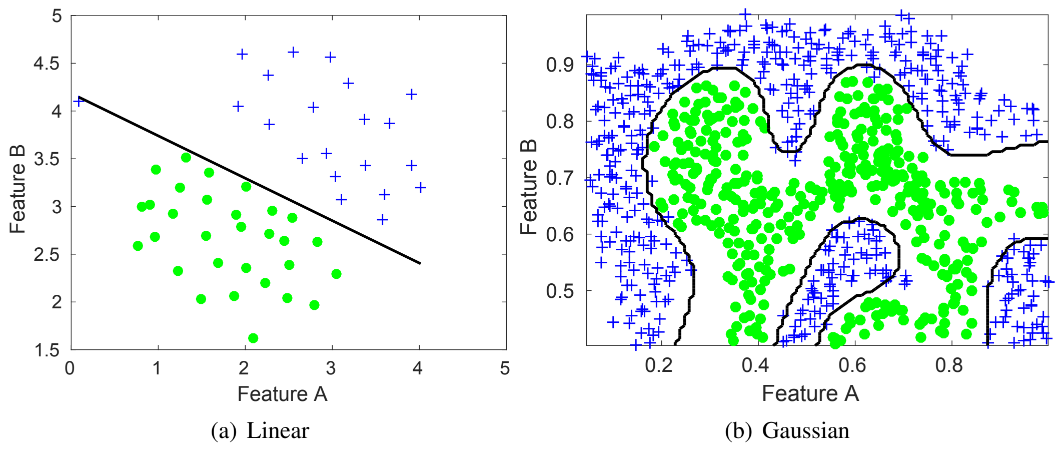


Figure 2.5: Examples of SVM, where different datasets are analyzed according to a linear (left) and a Gaussian (right) kernel.

2.2.3.3. Artificial neural networks

ANN is a supervised machine learning solution for both regression and classification. An ANN is a network of nodes, or *neurons*, grouped into three layers (input, hidden and output), which allows for nonlinear classification. Ideally, it can achieve zero training error.

Consider a training dataset $\{(\mathbf{x}_i, y_i) : \mathbf{x}_i \in \mathbb{R}^n, i = 1, \ldots, M\}$. Each hidden node h_l approximates a so-called logistic function in the form $h_l = 1/(1 + \exp(-\boldsymbol{\omega}_l \cdot \mathbf{x}))$, where $\boldsymbol{\omega}_l$ is a weight vector. The outputs of the hidden nodes are processed by the output nodes to approximate $\mathbf{y}$. These nodes use linear and logistic functions for regression and classification, respectively. In the linear case, the approximated output is represented as:

$$\hat{\mathbf{y}} = \sum_{l=1}^{L} h_l v_l = \sum_{l=1}^{L} \frac{1}{1 + \exp(-\boldsymbol{\omega}_l \cdot \mathbf{x})} v_l, \tag{2.10}$$

where L is the number of hidden nodes and v_l is the weight vector of the output layer. The training of an ANN can be performed by means of the *backpropagation* method that finds weights for both layers to minimize the mean squared error between the training labels y and their approximations $\hat{y}$. In the anticipatory networking literature, ANNs have been used for example to predict mobility in mobile ad-hoc networks [37, 181].

For both SVMs and ANNs no prior knowledge about the system is required but a large training set has to be acquired for parameter setting in the predictive model. A careful analysis needs to be performed while processing the training data in order to avoid both overfitting and underlearning.

2.2.4. Statistical Methods for Probabilistic Forecasting

Probabilistic forecasting involves the use of information at hand to make statements about the likely course of future events. In the following subsections, we introduce two probabilistic forecasting techniques: *Markovian models* and *Bayesian inference*.

2.2.4.1. Markovian models

These models can be applied to any system for which state transitions only depend on the current state. In the following we briefly discuss the basic concepts of discrete, and continuous time Markov Chains (MCs) and their respective applications to anticipatory networking.

A Discrete Time Markov Chain (DTMC) is a discrete time stochastic process $X_n (n \in \mathbb{N})$, where a state X_n takes a finite number of values from a set $\mathcal{X}$ in each time slot. The Markovian property for a DTMC transitioning from any time slot k to $k+1$ is expressed as follows:

$$P(X_{k+1} = j | X_k = i) = p_{ij}(k). \tag{2.11}$$

For a stationary DTMC, the subscript k is omitted and the transition matrix $\mathbf{P}$, where p_{ij} represents the transition probability from state i to state j, completely describes the model. Empirical measurements on mobility and traffic evolution can be accurately predicted using a DTMC with low computational complexity [32, 45, 47, 112, 136]. However, obtaining the transition probabilities of the system requires a variable training period, which depends on the prediction goal. In practice, the data collection period can be in the order of one [112] or even multiple weeks [46,81].

A DTMC assumes the time the system spends in each state is equal for all states. This time depends on the prediction application and can range from a few hundred milliseconds to predict wireless channel quality [73], to tens of seconds for user mobility prediction [45, 81], to hours for Internet traffic [112]. For tractability reason, the state space is often compressed by means of simple heuristics [46, 81, 117], K-means clustering [73, 136], equal probability classification [117], and density-based clustering [136].

Eq. (2.11) defines a first order DTMC and can be extended to the l-th order (i.e., transition probabilities depend on the l previous states). By Using higher order, DTMCs can increase the accuracy of the prediction at the expense of a longer training time and an increased computational complexity [45, 47, 136].

If the sojourn time of each state is relevant to the prediction, the system can be modeled as a Continuous Time Markov Chain (CTMC). The Markovian property is preserved in CTMC when the sojourn time is exponentially distributed, as in [48]. When the sojourn time has an arbitrary distribution, it becomes a Markov renewal process as described in [43, 44].

If the transition probabilities cannot be directly measured, but only the output of the system is quantifiable (dependent on the state), hidden Markov models allow to map the output state space to the unobservable model that governs the system. As an example, the inter-download times of video segments are predicted in [117], where the output sequences are the inter-download times of the already downloaded segments and the states are the instants of the next download request.

2.2.4.2. Bayesian inference

This approach allows to make statements about what is unknown, by conditioning on what is known. Bayesian prediction can be summarized in the following steps: 1) define a *model* that expresses qualitative aspects of our knowledge but has unknown parameters, 2) specify a *prior* probability distribution for the unknown parameters, 3) compute the *posterior* probability distribution for the parameters, given the observed data, and 4) make predictions by averaging over the posterior distribution.

Given a set of observed data $\mathcal{D} := \{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \dots, M\}$ consisting of a set of input samples $\mathcal{X} := \{\mathbf{x}_i \in \mathbb{R}^p : i = 1, \dots, M\}$ and a set of output samples $\mathcal{Y} := \{\mathbf{y}_i \in \mathbb{R}^q : i = 1, \dots, M\}$, inference in Bayesian models is based on the *posterior distribution* over the parameters, given by the *Bayes' rule*:

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{Y}|\mathcal{X})} \propto p(\mathcal{Y}|\mathcal{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}), \tag{2.12}$$

where $\boldsymbol{\theta}$ is the unknown parameter vector.

Two recent works adopting the Bayesian framework are [76] and [28]. The former focuses on spatial prediction of the wireless channel, building a 2D non-stationary random field accounting for pathloss, shadowing and multipath. The latter exploits spatial and temporal correlation to develop a general prediction model for the channel gain of mobile users.

2.2.5. Summary

Hereafter, we provide some guidelines for selecting the appropriate prediction methods depending on the application scenario or context of interest.

Applications and data – The predicted context is the most important information that drives decision making in anticipatory optimization problems (see Section 2.3). Thus, the selection of the prediction method shall take into consideration the objectives of the application and the constraints imposed by the available data.

Choosing the outputs – Applications define the properties of the predicted variables, such as dimension, granularity, accuracy, and range. For example, large granularity or high data aggregation (such as frequently visited location, social behavior pattern) is best dealt with similarity-based classification methods which provide sufficiently accurate prediction without the complexity of other model-based regression techniques.

System model and data – The application environment is equally important as its outputs, which determines the constraints of modeling. Often, an accurate analysis of the scenario might highlight linearity, deterministic and/or causal laws among the variables that can further improve the prediction accuracy. Moreover, the quality of dataset heavily affects the prediction accuracy. Different methods exhibit different level of robustness to noisy data.

Guidelines for selecting methods – To choose the correct tool among the aforementioned set, we study the rationale for adopting each of them in the literature and derive the following practical guidelines.

Model-based methods – When a physical model exists, model-based regression techniques based on closed-form expressions can be used to obtain an accurate prediction. They are usually preferable for long-term forecast and exhibit good resilience to poor data quality.

Time series-based methods – These are the most convenient tools when the information is abundant and shows strong temporal correlation. Under these conditions, time series methods provide simple means to obtain multiple scale prediction of moderate to high precision.

Causal methods – If the data exhibits large and fast variations, causality laws can be key to obtain robust predictions. In particular, if a causal relationship can be observed between the variables of interest and the other observable data, causal models usually outperform pure data-driven models.

Probabilistic models – If the physical model of the prediction variable is either unavailable or too complex to be used, probabilistic models offer robust prediction based on the observation of a sufficient amount of data. In addition, probabilistic methods are capable of quantifying the uncertainty of the prediction, based on the probability density function of the predicted state.

Table 2.3 characterizes each prediction method with respect to *properties of the context* and *constraints*. Note that the methods for predicting a multivariate process can be applied to univariate processes without loss of generality. The granularity of variables and the prediction range are described using qualitative attributes such as **S**hort, **M**edium, **L**arge, and **any** instead of explicit values. For example, for the time series of traffic load per cell, S, M and L time scales are generally defined by minutes, tens of minutes and hours, respectively, while for the time series of channel gain, they can be seen as milliseconds, hundreds of milliseconds and seconds, respectively. The sixth column reports the prediction type, that can be driven by **data**, **models** or **both**. Linearity indicates whether it is required (**Y**) or not (**N**) or applicable in **both** cases. The side information column states whether out-of-band information can (**both**), cannot (**N**) or must (**Y**) be used to build the model. Finally, the quality column reports whether the predictor is **weak** or **robust** against insufficient or unreliable dataset.

Table 2.3: Selected Prediction Methods: Variables of interest and constraints of modeling.

Prediction Method		**Properties of the Context**			**Constraints**			
Class	Methodology	Dimension	Granularity	Range	Type	Linearity	Side Info.	Quality
Time series	ARIMA	univariate	M/L	S	data	Y	N	weak
	Kalman filter	multivariate	M/L	S	data	Y	N	weak
	References	*ARIMA*: [6, 27, 28, 38, 67–69, 79, 93, 99, 125] *Kalman*: [30, 74]						
Classification	CF	multivariate	L	M/L	data	Y	both	robust
	Clustering	multivariate	L	M/L	data	both	both	robust
	Decision trees	multivariate	L	any	data	both	Y	robust
	References	*CF*: [50, 130, 132] *Cluster*: [33, 35, 78, 107, 114, 127–129] *Decision trees*: [36, 111, 113]						
Regression	Functional	multivariate	any	M/L	models	both	Y	robust
	SVM	multivariate	any	any	both	both	both	weak
	ANN	multivariate	any	any	data	both	both	weak
	References	*Functional*: [25, 26, 28, 66, 94, 95, 108] *SVM*: [78, 115, 133] *ANN*: [37, 77, 96, 97]						
Probabilistic	Markovian	multivariate	M/L	any	both	both	both	weak
	Bayesian	multivariate	any	any	both	both	Y	weak
	References	*Probabilistic*: [31, 32, 43–51, 70, 72, 81–83, 106, 112, 117, 126, 136] *Bayesian*: [4, 6, 40, 71, 80, 116, 122–124, 137, 139]						

2.3. Optimization Techniques for Anticipatory Networking

This section identifies the main optimization techniques adopted by anticipatory networking solutions to achieve their objectives. Disregarding the particular domain of each work, the common denominator is to leverage some future knowledge obtained by means of prediction to drive the system optimization. How this optimization is performed depends both on the ultimate objectives and how data are predicted and stored.

In general, we found two main strategies for optimization: (1) adopting a well-known optimization framework to model the problem and (2) designing a novel solution (most often) based on heuristic considerations about the problem. The two strategies are not mutually exclusive and often, when known approaches lead to too complex or impractical solutions, they are mixed in order to provide feasible approximation of the original problem.

Heuristic approaches usually consist of (1) algorithms that allow for fast computation of an approximation of the solution of a more complex problem (e.g., convex optimization) and (2) greedy approaches that can be proven optimal under some set of assumptions. Both approaches trade optimality for complexity and most often are able to obtain performance quite close to the optimal one. However, heuristic approaches are tailored to the specific application and are usually difficult to be generalized or to be adapted for different scenarios, thus they cannot be directly applied to new applications if the new requirements do not match those of the original scenario.

In what follows, we focus on optimization methods only and we will provide some introductory descriptions of the most relevant ones used for anticipatory networking. The objective is to provide the reader with a minimum set of tools to understand the methodologies and to highlight the main properties and applications.

2.3.1. Convex Optimization

Convex optimization is a field that studies the problem of minimizing a convex function over convex sets. The interested reader can refer to [155] for convex optimization theory and algorithms. Hereafter, we will adopt Boyd's notation [155] to introduce definitions and formulations that frequently appear in anticipatory networking papers.

The inputs are often referred to as the optimization variables of the problem and defined as the vector $\mathbf{x} = (x_1, \dots, x_n)$. In order to compute the best configuration or, more precisely, to optimize the variables, an objective is defined: this usually corresponds to minimizing a function of the optimization variables, $f_0 : \mathbb{R}^n \to \mathbb{R}$. The feasible set of input configurations is usually defined through a set of m constraints $f_i(x) \leq b_i$, $i = 1, \dots, m$, with $f_i : \mathbb{R}^n \to \mathbb{R}$. The general formulation of the problem is

$$\begin{aligned} \text{minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_i \leq b_i, \quad i = 1, \dots, m. \end{aligned} \tag{2.13}$$

The solution to the optimization problem is an optimal vector $\mathbf{x}^*$ that provides the smallest value of the objective function, while satisfying all the constraints.

The convexity property (i.e., objective and constraint functions satisfy $f_i(a\mathbf{x} + (1-a)\mathbf{y}) \le af_i(\mathbf{x}) + (1-a)f_i(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $a \in [0,1]$) can be exploited in order to derive efficient algorithms that allows for fast computation of the optimal solution. Furthermore, if the optimization function and the constraints are linear, i.e., $f_i(a\mathbf{x} + b\mathbf{y}) = af_i(\mathbf{x}) + bf_i(\mathbf{y})$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and $a, b \in \mathbb{R}$, the problem belongs to the class of *linear optimization*. For this class, highly efficient solvers exist, thanks to their inherently simple structure. Within the linear optimization class, three subclasses are of particular interest for anticipatory networking: least-squares problems, linear programs and mixed-integer linear programs.

Least-squares problems can be thought of as distance minimization problems. They have no constraints ($m = 0$) and their general formulation is:

$$\text{minimize} \quad f_0(\mathbf{x}) = ||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2, \tag{2.14}$$

where $A \in \mathbb{R}^{k\times n}$, with $k \ge n$ and $||x||_2$ is the Euclidean norm. Notably, problems of this class have an analytical solution $\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$ (where superscript ^{T} denotes the transpose) derived from reducing the problem to the set of linear equations $\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}$.

Linear programming (LP) problems are characterized by linear objective function and constraints and are written as

$$\begin{aligned} \text{minimize} \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}^T\mathbf{x} \le b, \end{aligned} \tag{2.15}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{n\times m}$ and $\mathbf{b} \in \mathbb{R}^n$ are the parameters of the problem. Although, there is no analytical closed-form solution to LP problems, a variety of efficient algorithms are available to compute the optimal vector $\mathbf{x}^*$. When the optimization variable is a vector of integers $x \in \mathbb{Z}^n$, the class of problems is called *integer linear programming* (ILP), while the class of *mixed-integers linear programming* (MILP) allows for both integer and real variables to co-exist. These last two classes of problems can be shown to be NP-hard (while LP is P complete) and their solution often implies combinatorial aspects. See [156] for more details on integer optimization.

In anticipatory networking, we find that resource allocation problems are often modeled as LP, ILP or MILP, by setting the amount of resources to be allocated as the optimization variable and accounting for prediction in the constraints of the problem. In [52], prediction of the channel gain is exploited to optimize the energy efficiency of the network. Time is modeled as a finite number of slots corresponding to the look-ahead time of the prediction. When dealing with multimedia streaming, the data buffer is usually modeled in the constraints of the problem by linking the state at a given time slot to the previous slot. The solver will then choose whether to use resources in the current slot or use what has been accumulated in the buffer, as in, e.g., [55]. Admission control is often used to enforce quality-of-service, e.g., [8, 107], with the drawback of introducing integer variables in the optimization function. In these cases, the optimal ILP/MILP formulation is followed by a fast heuristic that enables the implementation of real-time algorithms.

2.3.2. Model Predictive Control

Model Predictive Control (MPC) is a control theoretic approach that optimizes the sequence of actions in a dynamic system by using the process model of that system within a finite time horizon. Therefore, the process model, i.e., the process that turns the system from one state to the next, should be known. In each time slot t, the system state, $\mathbf{x}(t)$, is defined as a vector of attributes that define the relevant properties of the system. At each state, the control action, $\mathbf{u}(t)$, turns the system to the next state $\mathbf{x}(t+1)$ and results in the output $\mathbf{y}(t+1)$. In case the system is linear, both the next state and the output can be determined as follows:

$$\mathbf{x}(t+1) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) + \boldsymbol{\psi}(t) \tag{2.16}$$

$$\mathbf{y}(t) = \mathbf{C}\mathbf{x}(t) + \boldsymbol{\epsilon}(t), \tag{2.17}$$

where $\boldsymbol{\psi}(t)$ and $\boldsymbol{\epsilon}(t)$ are usually zero mean random variables used to model the effect of disturbances on the input and output, respectively, and $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are matrices determined by the system model.

At each time slot, the next N states and their respective outputs are predicted and a cost function $J(\cdot)$ is minimized to determine the optimal control action $\mathbf{u}^*(t)$ at $t = t_0$:

$$\mathbf{u}^*(t_0) = \arg\min_{\mathbf{u}(t_0)} J(\hat{\mathbf{x}}(t_0), \mathbf{u}(t_0)), \tag{2.18}$$

where $\hat{\mathbf{x}}(t_0)$ is the set of all the predicted states from $t = t_0 + 1$ to $t = t_0 + N$, including the observed state at $t = t_0$. The expression in (2.18) essentially states that the optimal action of the current time slot is computed based on the predicted states of a finite time horizon in the future. In other words, in each time slot the MPC sequentially performs a N step lookahead open loop optimization of which only the first step is implemented [157].

This approach has been adopted for on-line prediction and optimization of wireless networks [71, 99]. Since the process model (for the prediction of future states and outputs) is available in this kind of systems, autoregressive methods can be used along with Kalman filtering [99], or max-min MPC formulation [122]. In [71], Kalman filtering is compared to other methods such as mean and median value estimation, Markov chains, and exponential averaging filters.

Optimization based on MPC relies on a finite horizon. The length of the horizon determines the trade-off between complexity and accuracy. Longer horizons need further look ahead and more complex prediction but in turn result in a more foresighted control action [122]. Reducing the horizon reduces the complexity while resulting in a more myopic action. This trade-off is examined in [71] by proposing an algorithm that adaptively adjusts the horizon length. In general, the prediction horizon is kept to a fairly small size [99] to avoid high computation overhead.

It is worth noting that MPC methods can be extended to the nonlinear case. In this case, the prediction accuracy and control optimality increase at the cost of more complex algorithms to find the solution [157]. Another benefit of these approaches is their applicability to non-stationary problems.

2.3.3. Markov Decision Process

Markov Decision Process (MDP) is an efficient tool for optimizing sequential decision making in stochastic environments. Unlike MPCs, MDPs can only be applied to stationary systems where a priori information about the dynamics of the system as well as the state-action space is available.

A MDP consists of a four tuple $(\mathcal{X},\mathcal{U},\mathbf{P},r)$, where $\mathcal{X}$ and $\mathcal{U}$ represent the set of all achievable states in the system and the set of all actions that can be performed in each of the states, respectively. Time is assumed to be slotted and in any time slot t, the system is in state $x_t \in \mathcal{X}$ from which it can take an action u_t from the set $U_{x_t} \in \mathcal{U}$. Due to the assumption of stationarity, we can omit the time subscript for states and actions. Upon taking action u in state x, the system moves to the next state $x' \in \mathcal{X}$ with transition probability $\mathbf{P}(x'|x,u)$ and receives a reward equal to $r(x,u,x')$. The transition probabilities are predicted and modeled as a Markov Chain prior to solving the MDP and preserve the Markovian behavior of the system.

The goal is to find the optimal policy $\pi^* : \mathcal{X} \to \mathcal{U}$ (i.e., optimal sequence of actions that must be taken from any initial state) in order to maximize the long term discounted average reward $\mathbb{E}\left(\sum_{t=0}^{\infty}\gamma^t r(x_t,u_t,x_{t+1})\right)$, where $0 \leq \gamma < 1$ is called *discount factor* and determines how myopic (if closer to zero) or foresighted (if closer to 1) the decision process should be. In order to derive the optimal policy, each state is assigned to a value function $V^{\pi}(x)$, which is defined as the long term discounted sum of rewards obtained by following policy π from state x onwards. The goal of MDP algorithms is to find $V^{\pi^*}(x)(\forall x \in \mathcal{X})$. Given that the Markovian property holds, it has been proved that the optimal value functions follow the Bellman optimality criterion described below [158] :

$$V^{\pi^*}(x) = \max_{u\in\mathcal{U}} \sum_{x'\in\mathcal{X}'} \Big(r(x,u,x') + \gamma\mathbf{P}(x'|x,u)V^{\pi^*}(x')\Big)\,\forall x \in \mathcal{X}, \tag{2.19}$$

where $\mathcal{X}' \subset \mathcal{X}$ is the set of states for which $\mathbf{P}(x'|x,u) > 0$. In order to solve the above equation set, linear programming or dynamic programming techniques can be used, in which the optimal policy is derived by simple iterative algorithms such as policy iteration and value iteration [158].

MDPs are very efficient for several problems, especially in the framework of anticipatory networking, due to their wide applicability and ease of implementation. MDP-based optimized download policies for adaptive video transmission under varying channel and network conditions are presented in [70, 73, 126].

In order to avoid large state spaces (which limit the applicability of MDPs), there are cases where the accuracy of the model must be compromised for simplicity. In [126], a large video receiver buffer is modeled for storing video on demand but only a small portion of the buffer is used in the optimization, while the rest of the buffer follows a heuristic download policy. [70, 73] solve this problem by increasing the duration of the time slot such that more video can be downloaded in each slot and, therefore, the buffer is filled entirely based on the optimal policy. This, in turn, comes at the cost of lower accuracy, since the assumption is that the system is static within the duration of a time slot. Heuristic approaches are also adopted for on-line applications. For instance, creating decision trees with low depth from the MDP outputs is proposed in [73]. Simpler heuristics are also applied to the MDP outputs in [70, 126, 130].

If any of the assumptions discussed above does not hold, or if the state space of the system is too large, MDPs and their respective dynamic programming solution algorithms fail. However, there are alternative techniques to solve this kind of problems. For instance, if the system dynamics follow a Markov Renewal Process instead of a MC, a semi MDP is solved instead of the regular one [158]. In non-stationary systems, for which the dynamics cannot be predicted a priori or the reward function is not known beforehand, reinforcement learning [159] can be applied and the optimization turns into an on-line unsupervised learning problem. Large state spaces can be dealt with using value function approximation, where the value function of the MDP is approximated as a linear function, a neural network, or a decision tree [159]. If different subsets of state attributes have independent effects on the overall reward, i.e., multi user resource allocation, the problem can be modeled as a weakly coupled MDP [105] and can be decomposed into smaller and more tractable MDPs.

2.3.4. Game theoretic approaches

Although small in number, the papers adopting a game theoretic framework offer an alternative approach to optimization. In fact, while the approaches described in the previous subsections strive to compute the optimal solution of an often complex problem formulation, game theory defines policies that allow the system to converge towards a so-called equilibrium, where no player can modify her action to improve her utility. In mobile networks, game theory is applied in the form of matching games [135], where system players (e.g. users) have to be matched with network resources (e.g. base stations or resource blocks).

Three types of matching games can be used depending on the application scenario: 1) one-to-one matching, where each user can be matched with at most one resource (as in [80], which optimizes D2D communication in small cell scenarios); 2) many-to-one matching, where either multiple resources can be assigned to a single user (as in [116] for small cell resource allocation), or multiple users can be matched to a single resource (as in [41] for user-cell association); 3) many-to-many matching, where multiple users can be matched with multiple resource (as in [134] where videos are associated to caching servers).

Table 2.4: Optimization Methods Summary

Methodology	Properties of context	Modeling constraints
ConvOpt	Can support any context property, but larger system states slow the solver performance. The solution accuracy is linked to the context precision.	Linearity can be exploited to improve the solver efficiency, while data reliability impacts the solution optimality.
MPC	Usually offers the highest precision by coupling prediction and optimization.	The most computationally intensive technique.
MDP	Limited range and precision.	The most robust approach to low data reliability. Although the system setup can be computationally intensive, it allows for lightweight policies to be implemented.
Game theory	Limited granularity to allow the system to converge to an equilibrium.	Very low computational complexity. Fast dynamics hinder the system convergence.

2.3.5. Summary

This section (and Table 2.4) summarizes the main takeaways of this optimization handbook.

Convex Optimization methods – These methods are often combined with time series analysis or ideal prediction. The main reason is that they are used to determine performance bounds when the solving time is not a system constraint. Thus, convex optimization is suggested as a benchmark for large scale prediction. This may have to be replaced by fast heuristics in case the optimization tool needs to work in real-time. An exception to this is LP for which very efficient algorithms exist that can compute a solution in polynomial time. In contrast, convex optimization methods should be preferred when dealing with high precision and continuous output. They require the complete dataset and show a reliability comparable to that of the used predictor.

Model Predictive Control – MPC combines prediction and optimization to minimize the control error by tuning both the prediction and the control parameters. Therefore, it can be coupled with any predictor. The main drawback of this approach is that, by definition, prediction and optimization cannot be decoupled and must be evaluated at each iteration. This makes the solution computationally very heavy and it is generally difficult to obtain real-time algorithms based on MPC. The close coupling between prediction and optimization makes it possible to adopt the method for any application for which a predictor can be designed with the only additional constraint being the execution time. Objectives and constraints are usually those imposed by the used predictor.

Markov Decision Processes – MDPs are characterized by a statistical description of the system state and they usually model the system evolution through probabilistic predictors. As such, they best fit to scenarios that show similar objective functions and constraints as those of probabilistic predictors. Thus, MDPs are the ideal choice when the optimization objective aims at obtaining stationary policies (i.e., policies that can be applied independently of the system time). This translates to low precision and high reliability. Moreover, even though they require a computationally heavy phase to optimize the policies, once the policies are obtained, fast algorithms can easily be applied.

Game theory – Matching games prove to be effective solutions that, without struggling to compute an overly complex optimal configuration, let the system converge towards a stable equilibrium which satisfies all the players (i.e., no action can be taken to improve the utility of any player). These are the preferable solutions for those applications where the computational capability is a stringent constraint and where fairness is important for the system quality.

2.4. Issues, Challenges, and Research Directions

We conclude this survey by providing some insights on how anticipatory optimization will enable new 5G use cases and by detailing the open challenges of anticipatory networking in order to be successfully applied in 5G.

2.4.1. Context related analyses

Geographic context – Geographic context is essential to achieve seamless service. Depending on the optimization objective, a mobility state can be defined with different granularity in multiple dimensions (location, time, speed, etc.). For example, for handover optimization it is sufficient to predict the staying time in the current serving cell and the next serving cell of the user. Medium to large spatial granularity such as cell ID or cell coverage area can be considered as a state, and a trajectory can be characterized by a discrete sequence of cell IDs over time. State-space models such as Markov chains, HMM and Kalman filters fit the system modeling, while requiring large training samples and considerable insight to make the model compact and tractable. An alternative is the variable-order Markov models, including a variety of lossless compression algorithms (some of the most used belong to Lempel-Ziv family), where Shannon's entropy measure is identified as a basis for comparing user mobility models. Such an information-theoretic approach enables adaptive online learning of the model, to reduce update paging cost. Moving from discrete to continuous models, which are applied to assist the prediction of other system metrics with high granularity, e.g., link gain or capacity, regression techniques are widely used. To enhance the prediction accuracy, a priori knowledge can be exploited to provide additional constraints on the content and form of the model, based on street layouts, traffic density, user profiles, etc. However, finding the right trade-off between the model accuracy and complexity is challenging. An effective solution is to decompose the state space and to introduce localized models, e.g., to use distinct models for weekdays and weekends, or urban and rural areas.

Although mobility prediction has been shown to be viable, it has not been widely adopted in practical systems. This is because, unlike location-aware applications with users' permission to use their location information, mobile service providers must not violate the privacy and security of mobile users. To facilitate the next generation of user-centric networks, new interaction protocols and platforms need to be developed for enabling more user-friendly agreements on the data usage between the service providers and the mobile users.

Furthermore, next generation wireless networks introduce ultra-dense small cells and high frequencies such as mmWaves. The transmission range gets shorter and transmission often occurs in line-of-sight conditions. Thus, 2D geographic context with a coarse level of accuracy is not sufficient to fully utilize the future radio techniques and resources. This trend opens the door for new research directions in inference and prediction of 3D geographic context, by utilizing advanced feedback from sensors in user equipments such as accelerometers, magnetometers, and gyroscopes.

Link context – When predicting link context, i.e., channel quality and its parameters, linear time series models have the potential to provide the best tradeoff between performance and complexity. When the channel changes slowly, e.g., because users are static or pedestrian, it is convenient to exploit the temporal correlation of historic measurements of the users' channel and implement linear auto-regressive prediction. This can be quite accurate for very short prediction horizons and at the same time simple enough to be implemented in real time systems. Kalman filters can also be used to track errors and their variance, based on previous measurements, thus handling uncertainties. However, time series and linear models are not robust to fast changes. Therefore, in high mobility scenarios, more complex models are needed. One possible approach is to exploit the spatio-temporal correlation between location and channel quality. By combining the prediction of the channel qualities with the prediction of the user's trajectory, regression analysis, e.g., SVMs, can be employed to build accurate radio maps to estimate the long term average channel quality, which accounts for pathloss and slow fading, but neglects fast fading variations. Ideally, one should have two predictions available: a very accurate short term prediction and an approximate long term prediction.

Usually, such prediction is exploited to optimize the scheduling, i.e., resource allocation over time or frequency. Convex and linear optimization are often used when prediction is assumed to be perfect. In contrast, Markov models are applied when a probabilistic forecasting is available. Despite the great benefits that link context can potentially bring to resource (and more generally network) optimization, today's networks do not yet have the proper infrastructure to collect, share, process and distribute link context. Furthermore proper methods are needed not only to gather data from users, but also, to discard irrelevant or redundant measurements as well as to handle sparsity or gaps in the collected data.

Traffic context – Traffic and throughput prediction has a concrete impact on the optimization of different services of different networks at different time scales. Network-wide and for long time scales, linear time series models are already used to predict the macroscopic traffic patterns of mobile radio cells for medium/long-term management and optimization of the radio resources. At faster time scales and for specific radio cells or groups of radio cells, the probabilistic forecasting of the upcoming traffic, e.g., by using Markovian models, can be exploited to solve short-term problems including the radio resource allocation among users and the cell assignment problem.

Throughput prediction tools are then naturally coupled with video streaming services in mobile radio networks which have embedded rate adaptation capabilities. In this context, a good practice is to use simple yet effective look-ahead video throughput predictors based on time windows which are often coupled with clustering approaches to group similar video sessions. Deep learning techniques are also proposed to predict the throughput of video sessions, which offer improved performance at the price of a much higher complexity.

The data coming from traffic/throughput prediction can be effectively coupled with application/scenario-specific optimization frameworks. When targeting network-wide efficiency, centralized optimization approaches seem to be superior and more widely used. As an example, the problem of radio resource allocation in mobile radio networks is effectively representable and solvable though convex optimization techniques in semi-real-time scenario. In contrast, when the optimization has to be performed with the granularity of the technology-specific time slot, sub-optimal heuristics are preferable. Besides resorting to optimization approaches, control theoretic modeling is extremely powerful in all those cases where the optimization objective includes traffic (and queue) stability.

Social context – We can conclude that leveraging the social context of data transmission results in gains for proactive caching of multimedia content and can improve resource allocation by predicting the social behavior of users. For the former, determining the popularity of content plays a crucial role. Collaborative filtering is a well known approach for this purpose. However, due to the heavy tail nature of content popularity, trying to use this kind of models for a broad class of content will usually not lead to good results. However, for more specific and limited classes of content, i.e., localized advertisement, where a particular item is likely to be requested by a large number of users, popularity prediction is an appealing solution. In general, proactive caching requires that content is stored on caches close to the edge network in order not to put excessive load on the core network. For optimizing resource allocation using social behavior, the social interaction of different users can be used to create social graphs that determine the level of activity of each user and thereby make it possible to predict the amount of resources each user will need. Network utility maximization and heuristic methods are the most popular techniques for this context. Due to the complexity of modeling the social behavior of users, they are useful for wireless networks that either expose a great deal of measurable social interaction (device-to-device communication, dense cellular networks with small cells, local wireless networks in a sports stadium), or when resources are very scarce.

2.4.2. Anticipation-enabled use cases

Future networks are envisioned to cater to a large variety of new services and applications. Broadband access in dense areas, massive sensor networks, tactile Internet and ultra-reliable communications are only a few of the use cases detailed in [182]. The network capabilities of today's systems (i.e., 4G systems) are not able to support such requirements. Therefore, 5G systems will be designed to guarantee an efficient and flexible use (and sharing) of wireless resources, supported by a native software defined network and/or network function virtualization architecture [182]. Big data analysis and context awareness are not only enablers for new value added services but, combined with the power of anticipatory optimization, can play a role in the 5G technology.

Mobility management – Network densification will be used in 5G systems in order to cope with the tremendous growth of traffic volume. As a drawback, mobility management will become more difficult. Additionally, it is foreseen that mobility in 5G will be on-demand [182], i.e., provided for and customized to the specific service that needs it. In this sense, being able to predict the user's context (e.g., requested service) and his mobility behavior can be extremely useful in order to speed up handover procedures and to enable seamless connectivity. Furthermore, since individual mobility is highly social, social context and mobility information will be jointly used to perform predictions for a group of socially related individuals.

Network sharing – 5G systems will support resource and network sharing among different stakeholders, e.g., operators, infrastructure providers, service providers. The effectiveness of such sharing mechanisms relies on the ability of each player to predict the evolution of his own network, e.g., expected network load, anticipated user's link quality and prediction of the requested services. Wireless sharing mechanisms can strongly benefit from the added value provided by anticipation, especially when prediction is available at fine granularity, e.g., in a multi-operator scheduler [183].

Extreme real-time communications – Tactile Internet is only one of the applications that will require a very low latency (i.e., in the order of some milliseconds). Allocating resources and guaranteeing such low end-to-end delay will be very challenging. 5G systems will support such requirements by means of a new physical layer (e.g., a new air interface). However, this will not be enough if not combined with context information used to prioritize control information (e.g., used to move virtual or real objects in real time) over content [184]. Knowledge about the information that is transmitted and its specific requirements will be crucial in order to assign priorities and meet the expected quality-of-experience in a combined effort of physical and higher layers.

Ultra-reliable communications – Reliability is mentioned in several 5G white papers, e.g. in [182], as necessary prerequisite for lifeline communications and e-health services, e.g., remote surgery. A recent work [185] proposed a quantified definition of reliability in wireless access networks. As outlined here, a posteriori evaluation of the achieved reliability is not enough in order to meet the expected target, which in some cases is as high as 99.999%. To this end, it is mandatory to design resource allocation mechanisms that account for (and are able to anticipate the impact on) reliability in advance.

2.4.3. Open challenges

While the literature surveyed so far clearly points out how anticipatory networking can enhance current networks, this section discusses several problems that need to be solved for its wider adoption. In particular, we identified four functionalities that are going to play an important role in the adoption of anticipatory networking in 5G networks:

- **Measurements and information collection:** in order to provide means to obtain and share context information, future networks need to provide trusted mechanisms to manage the information exchange.

- **Data analysis and prediction:** information databases need interoperable procedures to make sure that processing and forecasting tools are usable with many possible information sources .

- **Optimization and decision making:** data and procedures are then exploited to derive system management policies.

- **Execution:** finally, in contrast to current procedures, anticipatory execution engines need to take into account the impact of the decisions made in the past and re-evaluate their costs and rewards in hindsight of the actual evolution of the system.

For instance, scheduling and load balancing are two processes that greatly profit from anticipatory networking and cannot be realized without a comprehensive integration of the four aforementioned functionalities in future generation networks. The realization of these functionalities poses the following important challenges.

Privacy and security – In our opinion, one of the main hindrances for anticipatory networking to become part of next generation networks is related to how users feel about sharing data and being profiled. While voluntarily sharing personal information has become a daily habit, many disapprove that companies create profiles using their data [186]. In a similar way, there might be a strong resistance against a new technology that, even though in an anonymous way, collects and analyzes users' behavior to anticipate users' decisions. Standards and procedures need to be studied to enforce users' privacy, data anonymity and an adequate security level for information storage. In addition, data ownership and control need to be defined and regulated in order to allow users and providers to interact in a trusted environment, where the former can decide the level of information disclosure and the latter can operate within shared agreements.

Network functions and interfaces – Many of the applications that are likely to benefit from anticipatory networking capabilities (i.e. decision making and execution) require unprecedented interactions among information producers, analyzers and consumers. A simple example is provided by predictive media streaming optimizers, which need to obtain content information from the related database and user streaming information from the user and/or the network operator. This information is then analyzed and fed to a streaming provider that optimizes its service accordingly. While ad hoc services can be realized exploiting the current networking functionalities, next generation applications, such as the extreme real-time communications mentioned above, will greatly benefit from a tighter coupling between context information and communication interfaces. We believe that the potential of anticipatory functionalities can be used in communication system and they could be applied to other domains, such as public transportation and smart city management.

Next generation architecture – 5G networks are currently being discussed and, while much attention is paid to increasing the network capacity and virtualizing the network functions, we believe that the current infrastructure should be enhanced with repositories for context information and application profiles [187] to assist the realization of novel predictive applications. As per the previous concerns above, sharing sensible information, even in an anonymized way, will require particular care in terms of users' privacy and database accessibility. We believe that anticipatory networking can potentially improve every kind of mobile networks: cellular networks will likely be the first to exploit this paradigm, because they already own the information needed to enable the predictive frameworks and it is only a matter of time and regulations to make it a reality. Once it will be integrated in cellular networks, other systems, such as public WiFi deployments, device-to-device solutions and the Internet of Things, will be able to participate in the infrastructure to exploit forecasting functionalities; in particular, we believe this will be applied to smart cities and multi-modal transportation.

Impact of prediction errors – When making and using predictions, one should carefully estimate its accuracy, which is itself a challenge. It might be potentially more harmful to use a wrong prediction than not using prediction at all. Usually, a good accuracy can be obtained for a short prediction horizon, which, however, should not be too short, otherwise the optimization algorithms cannot benefit from it. Therefore, a good balance between prediction horizon and accuracy must be found in order to provide gains. In contrast, over medium/long term periods, metrics can usually be predicted in terms of statistical behavior only. Furthermore, to build robust algorithms that are able to deal with uncertainties, proper prediction error models should be derived. In the existing literature, uncertainties are mainly modeled as Gaussian random variables. Despite the practicability of such an assumption, more complex error models should be derived to take into account the source (e.g., location and/or channel quality) as well as the cause (e.g., GPS accuracy and/or fast fading effect) of errors.

Chapter 3

Throughput Prediction for Mobile Network Users

The main contribution of this chapter is a novel synthetic model representing the impact of estimation and prediction errors on the bandwidth availability statistics to be able to study network resource optimization problems under forecasting uncertainties. In order for the model to account for the many different error sources, we analyze state of the art prediction models for both network resources as well as user mobility, which we subsequently organize in a taxonomy based on the time-scale and granularity of the prediction.

With reference to the survey of Chapter 2 we can classify this model at the boundary between geographic and link contexts and between time series and statistical analyses. In what follows, Section 3.1 provides an overview and taxonomy of predictors upon which our model is based. In Section 3.2, we discuss in detail the model for network resource availability under estimation and prediction errors. The model is applied to LTE cellular systems in Section 3.3.

3.1. Taxonomy of Predictors

In this section, we analyze predictors for both user mobility (3.1.1) and network resource availability (3.1.2) in order to understand the forecasting capability for mobile systems and the accuracy of the available solutions. The considered works cover a wide range of time scales, location granularities and levels of accuracy. To provide a comprehensive model, we classify them in three categories according to their time and space granularity.

The first group [101, 103, 104, 188], (1)-*net*, is the most coarse: network performance is modeled by analyzing the whole network at once, with a time scale on the order of minutes to hours; users are statistically mapped to base station cell ID or geographic location, i.e., predictions obtained by these models concern average throughput achievable in the location a given user is most likely to be found. Algorithms in the second group [35, 189–191], (2)-*cell*, combine user mobility information and network location specific information to refine prediction granularity.

Table 3.1: Prediction Taxonomy

Ref.	Cat.	Accuracy	Notes
[188]	(1)-net	$c_r \sim 0.8$	Provides a model for the number of user in a cell.
[101]		$\varepsilon \sim 0.15$	ARIMA models and wavelet MRA.
[103]		$\varepsilon \geq 0.01$	GARCH-ARIMA accurately models static high-speed network traffic.
[104]		$\varepsilon \in [0.01 - 1]$	Evaluates multi scale and s-sample prediction.
[189]	(2)-cell	$c_r \in [0.5 - 0.72]$	Compares Markovian (better) and Lempel-Ziv models.
[190]		$\varepsilon_l \sim 2$ m	User trajectory prediction.
[35]		$c_r \in [0.2 - 0.7]$	Route prediction on GPS data.
[191]		$c_r > 0.8$	Using pre-filtered data and Markov models. Prediction possible in the 98% of the cases.
[192]	(3)-user	$\varepsilon \in [0.05 - 2]$	Empirical study on user traces using wavelet approximations and filtering.
[193]		$\varepsilon \sim 1$	First attempt at mobile system bandwidth prediction.
[81]		n/a	Complete solution for mobile bandwidth forecast.
[112]		$\varepsilon \sim 0.01$	Spatial and temporal dynamics characterization of mobile Internet traffic.

Predictors belonging to this group aim at predicting the next cell a user is likely to visit, the congestion level in that cell and the time of the visit. Its timescale is between tens of seconds and a few minutes.

The third group [81,112,192,193], (3)-*user*, comprises the predictors with highest time granularity: in fact, most of the solutions in this group leverage filtering techniques and historical data. The aim, here, is to model the fast bandwidth variations experienced by the users on a timescale of tens of milliseconds up to a few seconds.

Table 3.1 groups the papers into the three categories and also provides a high level description of the papers. The "Cat." column specifies the name of the category, while the "Accuracy" column provides an evaluation of the effectiveness of the techniques. Here, we use the ratio between the mean square error of the prediction and the standard deviation of the original time series (e.g.: the user throughput, the bandwidth availability, etc.) $\varepsilon = \mathrm{MSE}(\tilde{x})/\sigma_x^2 = \sum_i (x_i - \tilde{x}_i)^2 / \sum_i (x_i - \mu_x)^2$, where x_i and $\tilde{x}_i$ are the i-th samples of the original time series and their predictions, respectively, and σ_x and μ_x are the standard deviation and the average of the original time series, respectively; c_r is the correct prediction rate defined as the ratio between the number of times the predicted location of a user is correct and the number of attempts; and ε_l represents the distance between the predicted and the correct user position.

3.1.1. Mobility Predictors

The most common methods to locate a mobile terminal are, in order of decreasing accuracy, the GPS, WiFi, and cellular network positioning. These solutions can identify a terminal's position with an average error on the order of 10, 100 and 500 meters, respectively [194].

Theoretical works, such as [195] and [42] studied characteristics of human behavior and found that an appreciable level of self-similarity exists among behavioral patterns and that, within due limits, forecasting is possible. Among the many studied properties, we highlight the one asserting that the probability of a user to be found in a given location is approximately inversely proportional to the location rank.

Some predictors aim at estimating the next user position on a grid representing network cells: $[189]_{(2)}$ compares Markovian and Lempel-Ziv models trained with the sequences of locations a user visited in the past, while $[188]_{(1)}$ studies the accuracy of mobility modeling. Notably, the first paper comes to the conclusion that second order Markov models provide a good trade off between complexity and accuracy achieving a correct prediction rate $c_r \in [0.5 - 0.72]$ on mobility traces collected from more than 6000 users of Dartmouth College's wireless network. The second paper provides an effective way to estimate the number of users in a cell and, consequently, the congestion level.

Other predictors deal with routes and trajectories. $[190]_{(2)}$ uses 1-sample predictions of user position to improve the performance of a routing protocol. (An s-sample prediction computes the first s unknown samples of a given time series.) The location prediction accuracy is claimed to be on the order of a few meters, with a position error $\varepsilon_l \sim 2$ m. The work in $[35]_{(2)}$ focuses on predicting complete routes from historical GPS data and obtains a $c_r \in [0.2 - 0.7]$. Here, the best results are obtained when excluding single trips from the dataset.

Finally, $[191]_{(2)}$ uses second and third order Markov models trained on a pre-filtered leap graph to model and predict cellular user mobility. The solution is able to achieve a $c_r \geq 0.8$ in 98% of the cases. Finally, recent works, such as [196] and $[81]_{(3)}$, directly exploit position information obtained from navigation systems to map bandwidth availability to locations. While these solutions provide a prediction that is based on the actual intended destination of the user, the accuracy of the prediction is still limited by the accuracy of the positioning system and the possibility of user detours. To the best of our knowledge, a detailed study linking location prediction accuracy to bandwidth/throughput prediction accuracy does not exist.

3.1.2. Bandwidth Predictors

One of the most relevant studies on traffic dynamics for cellular networks is [109], which conducted the first detailed wide scale analysis of network usage and subscriber behavior. The paper characterizes mobility and temporal activity patterns and identifies their relation to traffic volume. Traffic has been analyzed from the base station point of view, identifying its variations over space and time.

Earlier works such as $[101]_{(1)}$, $[192]_{(3)}$ and $[103]_{(1)}$ studied different filtering techniques, namely MEAN, LAST, MA, AR, ARMA, ARIMA, and AutoRegressive Fractionally Integrated Moving Average (FARIMA), all of which are different combinations of moving average and autoregressive filtering. We refer the interested reader to the source papers for the details. Although different papers use slightly different metrics, the following conclusions can be drawn:

Table 3.2: MCS coefficients

Modulation	N/A	QPSK						16QAM			64QAM					
CQI	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
G_i	$-\infty$	-6.00	-4.14	-2.29	-0.43	1.43	3.29	5.14	7.00	8.86	10.71	12.57	14.43	16.29	18.14	20.00
c_i	0	0.15	0.23	0.38	0.60	0.88	1.18	1.48	1.91	2.41	2.73	3.32	3.90	4.52	5.12	5.55

low order filtering techniques coupled with smoothing solutions (e.g., wavelet MultiResolution Analysis (MRA) or wavelet approximation) are able to provide 1-sample static network traffic predictions with an error as low as $\varepsilon = 0.05$ and almost always lower than the variance of the original signal, $\varepsilon = 1$. (In the latter case, the predicted sample error would be as large as those that would have been obtained by generating random samples from a distribution with the same variation as the original signal). The error decreases with larger timescale and smoother approximation of the signal.

Subsequent work in [104]$_{(1)}$ compares FARIMA and GARCH filtering techniques in terms of both time scale and the number of predicted samples s. Results obtained from Internet traffic traces show that GARCH outperforms FARIMA, achieving an error that is four time smaller. The authors confirm that the error decreases with increased signal timescales and increases with the number of predicted samples s. In particular, the error becomes as high as the variance of the original signal for $s = 10$ and $s = 100$ samples for FARIMA and GARCH, respectively. Also, GARCH errors are slightly smaller than half the variance for $s = 10$ samples and beyond.

[193]$_{(3)}$ and [112]$_{(3)}$ study resource availability in mobile systems. The former observes no significant correlation within a single trip, but throughput traces show a higher degree of self-similarity during repeated trips. The latter paper, instead, classifies traffic according to spatial features and proposes a multi-class model to predict traffic, achieving promising results ($\varepsilon \sim 0.01$). Finally, although standard filtering techniques for static environments are less effective when applied to throughput of mobile nodes, they provide better accuracy when location is used as a context.

3.2. Bandwidth Availability Model

Based on the previous taxonomy, this section determines the main error sources and their impact on the statistical distribution of the predicted throughput. Fig. 3.1 shows examples of effects of errors on throughput prediction: the x-axis represents how far into the future the prediction is made, while the y-axis represents the predicted throughput and the corresponding estimation error. Note that purpose of the figure is to graphically exemplify the predictor categories; it is primarily intended to provide an intuition.

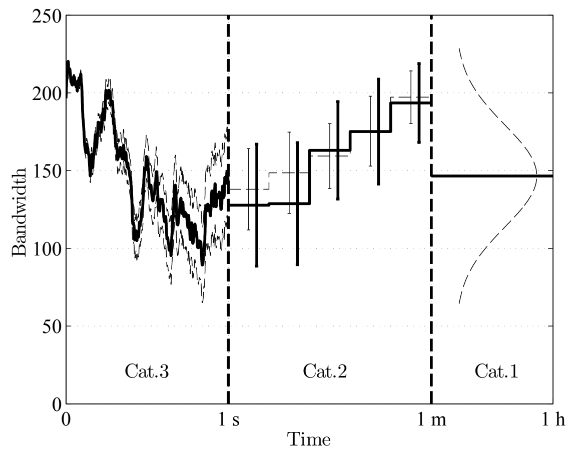


Figure 3.1: Bandwidth forecasting examples: category 3, 2 and 1 predictor outputs are shown on the left hand side, in the center and on the right hand side, respectively.

The figure examines the three categories of the taxonomy starting from (3)-*user* category on the left hand side. Here, the solid line represents the prediction itself, while the two dashed lines represent the confidence range of the prediction. Although the accuracy degrades with time, predictors belonging to this category are able to closely follow the throughput variations. As soon as the confidence range becomes as large as the signal's standard deviation, category (2)-*cell* predictors becomes as effective as category (3)-*user* predictors. In the center, predictions obtained from the category (2)-*cell* are shown. Here, the predictions are averaged over longer time periods and their variability is represented by error bars. The solid line represents the actual prediction average along with its standard deviation, while the dashed line represents the same for the original signal. Predictors in this category infer user throughput from their position and statistics of the corresponding network cell. Whenever it is not possible to predict the next user location, only predictors in category (1)-*net* can be used (right hand side of Fig. 3.1). They derive an estimate of user throughput from general network information using, for example, the generic distribution of user throughput in the overall network (shown in the figure as a dashed line). To model the impact of errors on the predictors, we start from a simple formulation of the phenomenon itself. A very popular user throughput model can be found, for instance, in [197]. Here, the throughput T of a user with a distance of d kilometers to the transmitter and competing with N other users uniformly distributed within the coverage area of the transmitter, is represented as a function of the SINR γ, and N:

$$T = g_T(\Gamma, N) = T_0\eta/N, \tag{3.1}$$

where $\Gamma = 10\log_{10}\gamma$ is the SINR in dB, T_0 is a parameter specific to the actual cellular system and $\eta = g_\eta(\Gamma)$ is the spectral efficiency for that SINR. The SINR is a function of d and the fast fading gain r:

$$\gamma = g_\gamma(d, r) = \gamma_0 r/d^\alpha, \tag{3.2}$$

where γ_0 is a technology specific parameter and α is the pathloss exponent.

For what concerns errors themselves, different predictors are impacted by different error sources: for instance, those belonging to the third category try to model the short term behavior of the achievable throughput starting from past information. Thus, predicted throughput $\tilde{T} = T + e_T$, is the sum of the actual throughput and the prediction error. Given that the error e_T has a probability density function (pdf) $f_{e_T}(e)$, the predicted throughput will have a pdf $f_{\tilde{T}} = f_{e_T}(e - T)$. Also, in the worst case the s-sample prediction can be modeled has the sum of s i.i.d random variables with distribution $f_{e_T}(e)$. Thus the s-sample predicted throughput distribution can be obtained as $f_{\tilde{T}(s)} = f_{e_T}((e - T)/s)/s$, which will have an expected value $\mu_{\tilde{T}(s)} = T(s) + s\mu_{e_T}$ and standard deviation $\sigma_{\tilde{T}(s)} = s\sigma_{e_T}$. Note that increasing s makes the prediction less and less accurate up to a point where the standard deviation of the prediction becomes comparable to the variability of the throughput σ_T.

Beyond this point using this type of predictors is useless and category 2 and category 1 predictors should be used. In this case, most of the predictors try to first estimate system parameters, such as the distance d and the number N of users and, from those, estimate the throughput distribution. Thus, in order to model the latter from the distributions of d and N, we will proceed as follows. First we analyze the distribution of the SINR given that N user are competing for the channel. It depends on the joint distribution $f_{r,d}(r, d|N)$, of the fading gain r and the distance d according to (3.2):

$$f_\gamma(\gamma|N) = \int_0^\infty f_{r,d}(g_\gamma^{-1}(\gamma, d), d|N) \left| \frac{\partial g_\gamma^{-1}(\gamma, d)}{\partial \gamma} \right| \mathrm{d}d, \tag{3.3}$$

where $g_\gamma^{-1}(\gamma, d)$ is the inverse function of (3.2) and we remove the variable d from the joint distribution $f_{\gamma,d}(\gamma, d|N)$ by integrating it on its whole support. Note that it is important to condition on N in order to account for opportunistic gain effects.

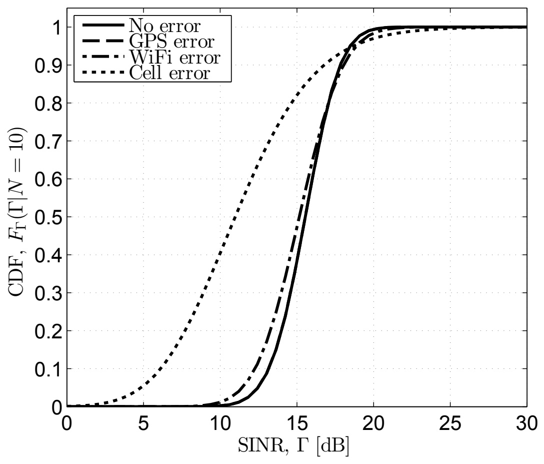
The last step requires to compute the throughput from the SINR using $g_\eta(\Gamma)$, which can be a piece-wise constant or other non-differentiable functions. In this case it is easier to use the CDF, since we can avoid to use the derivative. In fact, the throughput CDF $F_T(x|N) = P(T \leq x) = P(g_T(\gamma) \leq x) = P(\gamma \leq g_T^{-1}(x)) = F_\gamma(\gamma^*|N)$, where $\gamma^* = g_T^{-1}(x)$. Thus,

$$F_T(x|N) = \int_0^{\gamma^*} f_\gamma(\gamma|N) \mathrm{d}\gamma. \tag{3.4}$$

The SINR and the throughput distributions can be obtained removing the dependency on N by multiplying by the probability mass function (pmf) of the number of user p_N, and summing over N. Thus,

$$F_\gamma(\gamma) = \sum_{i=1}^{M_N} p_i \int_0^\gamma f_\gamma(\gamma|i), \tag{3.5}$$

$$F_T(x) = \sum_{i=1}^{M_N} p_i F_T(x|i), \tag{3.6}$$



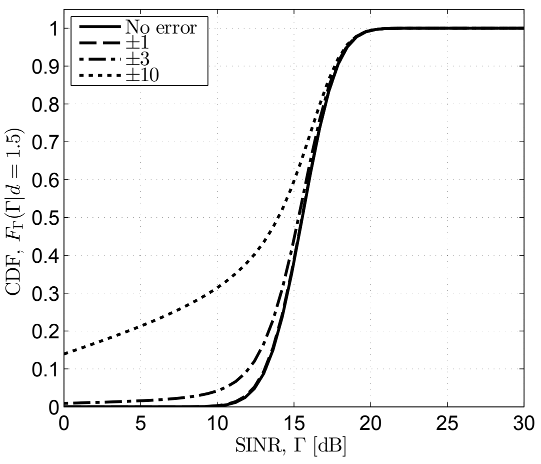


Figure 3.2: Plots of the SINR CDF F_Γ, given a perfect knowledge of $N = 10$ (left) or a perfect knowledge of $d = 1.5$ Km (right). In the former case the standard deviation σ_d, of the distance is set as that of the most common localization systems, while in the latter $\sigma_N \in \{0, 1, 3, 10\}$.

where M_N is chosen so that $p_{M_N} > 0$ and $p_{M_N+i} = 0, \forall i > 0$. Note that, thanks to the independence of the fading and the distance distributions, their joint distribution can be written as the product of the two distributions:

$$f_{r,d}(r, d|N) = f_r(r|N)f_d(d). \tag{3.7}$$

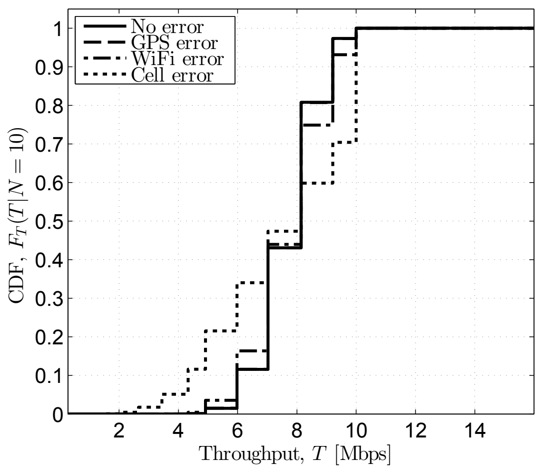
It is easy to customize the model by modifying the distributions of three basic random variables, namely p_N, $f_d(d)$, $f_r(r|N)$. In particular, it is possible to include temporal and/or spatial dependencies by letting the distributions vary according to the location and the time.

3.3. Results

In this section we apply the model to the case of an LTE cellular system as defined in [198] adopting a PF scheduler modeled according to the results in Section II.D and III.B in [197].

In particular, we provide more specific definitions for some of the previous parameters: $T_0 = N_R B_R$, where N_R is the number of resource blocks and B_R is channel bandwidth; $\gamma_0 = 10^{(P_T - N_f + C)/10}$, where P_T is the eNodeB transmission power in dB N_f is the noise plus interference power in dB and $C = 128.1$ dB is a constant modeling other effects (such as antenna gains, frequency dependency, etc.); $g_\eta(\Gamma) = c_i$ if $G_i < \Gamma \leq G_{i+1}$ with $i \in \{0, \ldots, 15\}$ and $G_{16} = \infty$. c_i is the bit efficiency of the modulation of the i-th Modulation and Coding Scheme (MCS). The values for c_i and G_i used in the paper are derived from [198] and are given in Table 3.2.

In order to derive the exact expression for the SINR and the throughput distributions, we need to specify the distributions for the fading gain r, the distance d, between the user equipment and



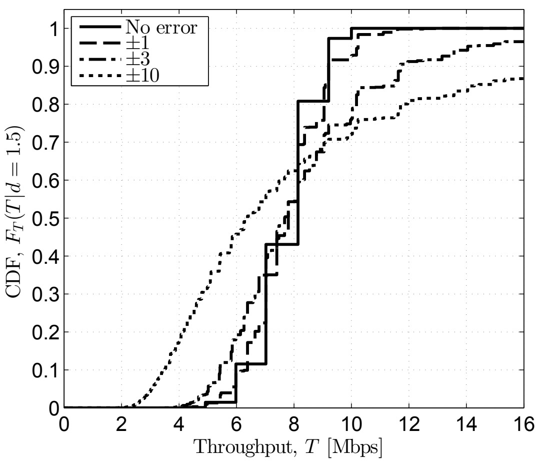


Figure 3.3: Plots of the throughput CDF F_T, given a perfect knowledge of $N = 10$ (right left side) or a perfect knowledge of $d = 1.5$ Km (right hand side). In former case the standard deviation σ_d, of the distance is set as that of the most common localization systems, while in the latter, $\sigma_N \in \{0, 1, 3, 10\}$.

the eNodeB, and the number N, of user in the cell. For what concerns the fading gain, in this paper we follow the results of [197], which models the opportunistic gain obtainable by the PF scheduler as follows:

$$f_r(r|N) = N(1 - e^{-r})^{N-1}e^{-r}. \tag{3.8}$$

This gain is associated to the higher probability for a user to be scheduled having a high SINR, when more users are competing for the channel.

The distance distribution $f_d(d)$, is obtained as the sum of two components: the actual distance distribution and the error committed in evaluating and/or predicting it. In the following, we analyze the case of a static user, whose distance is obtained with the three most common methods: GPS, WiFi and cell signal strength. In all the three cases we model the distance with a Gaussian distribution with an average $\mu_d = d^*$, equal to the correct user position d^*, and a standard deviation $\sigma_d = \{10, 100, 500\}$ meters, for GPS, WiFi and cell localization [194], respectively. Since the Gaussian distribution can lead to positive probability for negative values, we will normalize by $1 - \Phi(-\mu_d/\sigma_d)$, where $\Phi(x)$ is the CDF of a Gaussian distribution computed in x.

Similarly, the distribution of the number of users N, depends both on the actual value N^*, and the estimation error. As above, we take the Gaussian distribution as a reference:

$$p_i = \Phi((i - \mu_N)/\sigma_N)/(\sigma_N \sum_j p_j) \tag{3.9}$$

with $i \in \{1, \ldots, M_N\}$, where $\mu_N = N^*$ is the average value of the distribution and $\sigma_N \in \{0, 1, 3, 10\}$ are the standard deviation values we studied in the following examples of Fig. 3.2 and Fig. 3.3.

In these two examples, we focused on a single error at a time so as to separate the effects of an erroneous knowledge of N and d. The plot presents results for which we applied the aforementioned distributions to (3.3), obtaining:

$$f_\gamma(\gamma|N) = \int_0^\infty \frac{Nd^\alpha}{\gamma_0\sigma_d}\Big(1 - e^{-\frac{\gamma d^\alpha}{\gamma_0}}\Big)^{N-1} e^{-\frac{\gamma d^\alpha}{\gamma_0}} \phi\Big(\frac{d-\mu_d}{\sigma_d}\Big)\mathrm{d}d. \tag{3.10}$$

Now it is possible to compute γ^* as

$$\begin{aligned} \gamma^* &= 10^{g_\eta^{-1}(\frac{TN}{N_R B_R})/10} \\ \gamma_{i,N} &= 10^{G_{i+1}/10} \text{ with } \frac{c_i T_0}{N} \leq T < \frac{c_{i+1} T_0}{N}, \end{aligned} \tag{3.11}$$

which depends on both the bandwidth and the number of users. Now it is possible to compute (3.5) and (3.6), by using (3.10), (3.11) and (3.9).

In particular, Fig. 3.2 (left) shows $F_\Gamma(\Gamma|N = 10)$, using $f_d(d) = \mathcal{N}(\mu_d = d^* = 1.5, \sigma_d)$, and $\sigma_d \in \{0, 0.01, 0.1, 0.5\}$ to represent a static user, whose position is obtained with a localization error ranging from perfect knowledge to the worst approximation of a cell system localization. The figure shows that only with the precision of GPS is it possible to accurately estimate the statistical distribution of the SINR and that, if GPS information is lacking, the SINR prediction distribution becomes very wide even for static users.

Similarly, Fig. 3.2 (right) shows $F_\Gamma(\Gamma|d = 1.5)$ and the number of users distributed according to (3.9) using $\mu_N = 10$, $\sigma_N \in \{0, 1, 3, 10\}$ and $M_N = \mu_N + 5\sigma_N$. Again, for low σ_N, the distribution maintains the original shape, but as soon as $\sigma_N > 1$ the SINR distribution starts to get wider and is shifted towards the left. Note that, an error on N implies that $P(\tilde{\gamma} > \gamma) = 0\ \forall\gamma$, which is a direct consequence of the modeling of the opportunistic gain of the PF scheduler.

The last two figures, Fig. 3.3 (left) and Fig. 3.3 (right), study F_T with errors on d and N, respectively. The error distributions are shaped as above, but this time the discontinuities of $g_\eta(\gamma)$ are evident. In particular, for a wider SINR distribution a larger number of MCS get positive probability of being used. Also, on the right hand side figure, the throughput CDF becomes smoother and smoother for increasing σ_N. This is due to the wider range of $\gamma_{i,N}$ introduced by (3.11). Besides the trivial conclusion that the throughput distribution widens as the uncertainties grow, our model allows to compute where the correct value of the throughput is more likely to be found when a given prediction is computed. Also, the model allows to estimate the likelihood of the throughput to fall below a given threshold, thus enabling the study of resource allocation techniques when future information has limited reliability.

Chapter 4

Modeling Throughput Prediction Errors as Gaussian Random Walks

In the previous chapter, we analyzed the state of the art in mobile throughput and user mobility prediction in order to derive a composite model for prediction error. Short term prediction, shown in Fig. 4.1 on the left, is most often based on time series filtering techniques [103, 192], while medium and long term prediction, shown on the right, is usually derived from mobility aspects and networks dynamics.

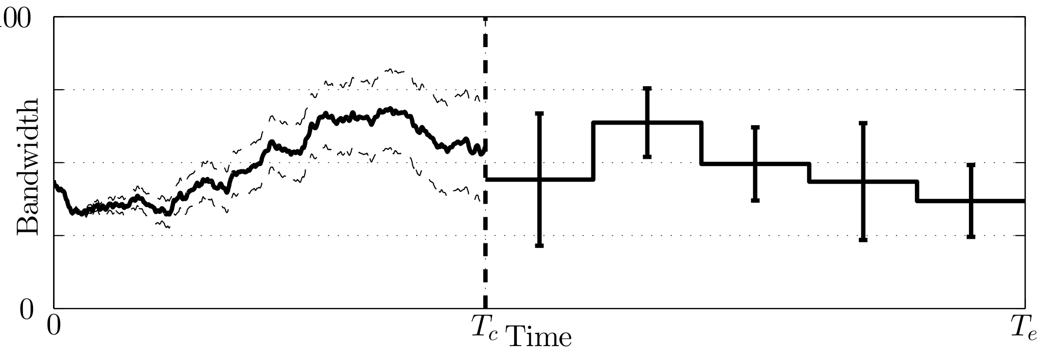


Figure 4.1: Capacity availability prediction uncertainties: short term predictors are useful until time T_c, while the medium term model is used until slot T_e.

Here, we will focus on predictors represented in the left part of the figure, where the solid line represent a possible trace of throughput evolution. The dashed lines show the boundary of the region the prediction is likely to fall in. The short term predictors start to be useless at time T_c when the prediction error is as big as that obtained by randomly drawing the next samples from the statistic distribution of the original phenomenon. Statistical distribution can be used from T_c until T_e, the time when no statistical considerations can be derived from mobility prediction.

In particular, we model the short term prediction error of a Gaussian random walk, which provides a close fit to the original random process and allows for a simple mathematical analysis of the impact of imperfect prediction on network optimization. In Section 4.1 we discuss the model use to derive mobile networks characteristics and the filtering technique we used for prediction. Section 4.2 gives details on prediction error and present our Gaussian random walk model.

4.1. System model

This section focuses on the prediction of downlink rate between base station (eNodeB) and UE. User throughput in mobile networks depends on several aspects and it is most always modeled as a function of the SINR γ and the number of active users in the cell K. The SINR is usually modeled as a function of the distance d between eNodeB and UE, the K active users and the scheduler type. In what follows we specifically address LTE technology and the related throughput model proposed by Østerbø [197] for the case of proportional fair scheduler and the K users uniformly distributed in the cell coverage area. According to this model, the throughput g can be expressed as:

$$g = \eta(\gamma_0 d^{-\alpha} r(K))/K, \tag{4.1}$$

where $\eta(x)$ is a piece-wise constant function associating throughput to SINR ranges, γ_0 is a constant scaling factor related to environmental and system parameters (e.g., transmit power, antenna gains, etc.), $\alpha \in [2,4]$ is the exponent of the pathloss law and $r(K)$ is the fast fading gain and depends on K to model opportunistic gain achieved by the scheduler. A throughput value obtained from Eq. (4.1) has a coherence time T_f which is inversely proportional to the user movement speed s [199]. Thus, we average $\lceil T_s/T_f \rceil$ throughput values to filter fast fading variations.

In order to obtain user movement traces we let the user move with constant speed and direction in an area where cells are randomly placed. In particular, the position of each eNodeB along the user path is chosen so that the maximum distance between the UE and the closest eNodeB is never larger than a given communication range. Every T_s seconds the UE-eNodeB distance is measured as the distance between the UE and the closest eNodeB in the area. The eNodeB random placement is equivalent to assume random variation in the user speed and constant distance between eNodeBs. In order to contain the dimensionality of the problem, in this paper we only study movement sequences characterized by constant speed and direction.

For any given tuple of parameters (s, T_s) we can generate any number of sequences $D(s, T_s) = d_i, i \in [1, T_l]$ of any length T_l. Subsequently, we can generate throughput sequences $G(s, T_s, K) = g_i, i \in [1, T_l]$, where g_i is obtained by averaging $\lceil T_s/T_f \rceil$ values obtained from d_i through Eq. (4.1). For what concerns prediction itself, we limited our focus on ARMA filters. We choose this technique, because it is well studied and it is simple to implement in mobile phones. The basic ARMA model is as follows:

$$X_i = c + \varepsilon_t + \sum_{j=1}^{p} \varphi_j X_{i-j} + \sum_{k=1}^{q} \theta_k \varepsilon_{i-k}, \tag{4.2}$$

where c is a constant, ε_i are white noise error terms, φ_j, θ_k, p and q are the autoregressive and the moving average coefficients and their respective orders and X_i is the reference signal. This model is referred to as an ARMA(p, q) with reference to the order of the two parts of the filter. To determine the order to be used, we followed the Box-Jenkins method [200] using automatic inspection of autocorrelation and sampled partial autocorrelation functions.

To generate error sequences and their statistics we operate as follows: first we obtain the optimal order of the ARMA filters to be used and, for each tuple of speed and sampling period (s, T_s), we generate a single very long training sequence $G_T(s, T_s, K)$ from which we tune the filter coefficients; subsequently, we generate shorter throughput sequences $G_i(s, T_s, K), i \in [1, 100]$ to test the filter on. In particular, we obtain filters $F(s, T_s, K)$ from $G_T(s, T_s, K)$ and we use the filters to predict the sequences $\tilde{G}_{ij}(s, T_s, K) = \tilde{g}_{ijk}, i \in [1, 100], j \in [1, 100], k \in [\max\{p, q\} + j, \max\{p, q\} + T_p + j]$ or, in other words, from each of the 100 sequences we generate 100 predicted sequences starting at different points and long T_p values. Finally, we compute errors $e_{ijk} = \tilde{g}_{ijk} - g_{ik}$ and the error sequences $E_{ij}(s, T_s, K)$ from which we further obtain the sequences $\sigma_k^2(s, T_s, K) = \mathrm{E}[(e_{ijk} - \mu)^2]/\sigma_G^2$, which represent the variance of the k-th prediction error normalized to the variance σ_G^2 of the original training signal $G_T(s, T_s, K)$.

4.2. Prediction error model

This section proposes to use a Gaussian random walk to approximate the sequences $E_{ij}(s, T_s, K)$. Gaussian random walks are interesting, because their total variance at time t is proportional to the interval duration and they can be expressed as a sum of i.i.d Gaussian random variables.

Before approaching the fitting of the model itself, we verified that assuming the error sequences to be drawn from zero mean normal distribution was a valid hypothesis. To do so, we perform the Kolmogorov-Smirnov [201] test between the generated error sequences and theoretical normal distributions with zero mean and the same variance as the error sequences. All the tests performed rejected the null hypothesis according to which the error and the normal distributions are not equal.

Subsequently, by visual inspection of the $\sigma_k^2(s, T_s, K)$ we noticed that: *i*) it increases with the prediction distance k, *ii*) the steepness is increasing with both s and T_s; *iii*) the minimum error is decreasing with both s and T_s; *iv*) T_c can be obtained as the minimum k so that $\sigma_k^2(s, T_s, K) = 1$; *v*) the number of active users K has a negligible impact on the prediction error.

Thus we are looking for a family of linear equations that approximates the variance sequence:

$$\sigma_k^2(s, T_s) = \begin{cases} A(s, T_s)k + B(s, T_s) & k \leq T_c/T_s \\ 1 & \text{otherwise} \end{cases}, \tag{4.3}$$

where $A(s, T_s)$ represent the steepness and $B(s, T_s)$ the offset of the process or, in other words, how fast the prediction reliability decreases and how large is the intrinsic randomness of the

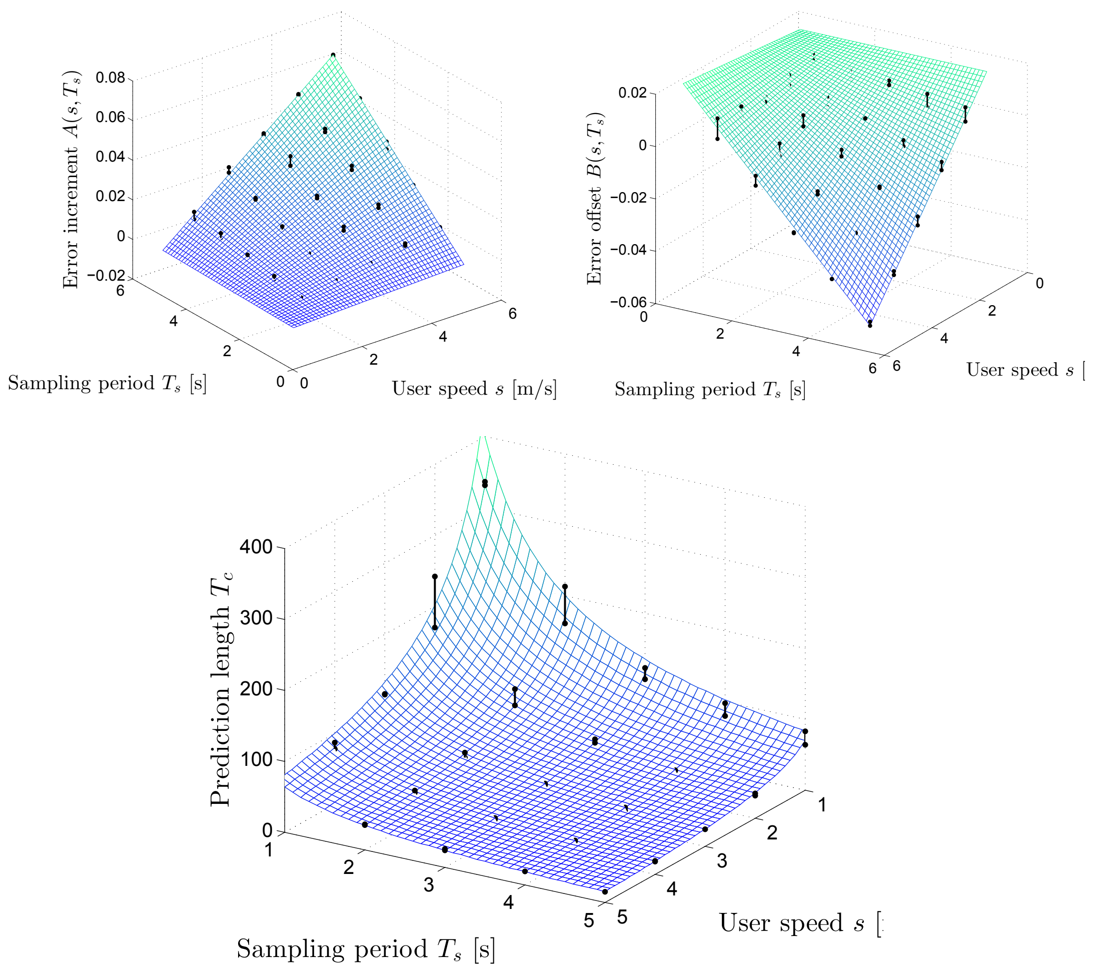


Figure 4.2: Comparison between the collected data and the fitted model varying s and T_s. Starting from the left, the figures show the approximation of the A (left), B (center) and T_c (right) parameters as surfaces and the distance from the surface to the actual data as lines.

process respectively. According to *ii*) and *iii*) we fit two linear functions on the sT_s product to approximate A and B respectively and we obtain:

$$\begin{aligned} A(s,T_s) &= A_1 s T_s + A_2 \\ B(s,T_s) &= B_1 s T_s + B_2. \end{aligned} \tag{4.4}$$

Fig. 4.2 shows how close the model fits the data. The model coefficients have been obtained from the original sequences by imposing a perfect match for $s = 1$ and $T_s = 1$ and minimizing the least square error in the other points. In particular, Fig. 4.2(a) and 4.2(b) show the surfaces obtained from Eq. (4.4) and the distance from the actual data and the surfaces. Also, Fig. 4.2(c) show the prediction validity length derived from the model $T_c = T_s(1 - B(s,T_s))/A(s,T_s)$ (surface) compared to the same obtained from the data.

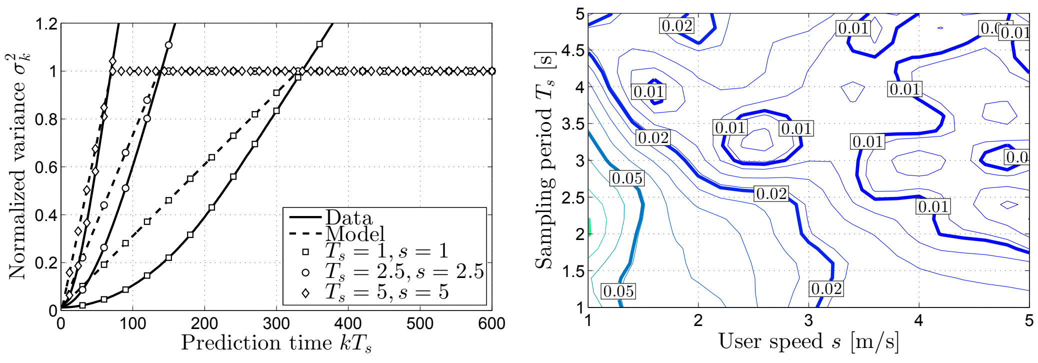


Figure 4.3: On the left a comparison between model and data for different s, T_s couples. On the right a contour plot of the average distance between the model and the data.

Fig. 4.3(a) visualizes how the model fits the data for a few speed-sampling time couples: solid and dashed lines represent the normalized variance σ_k^2 obtained from the data and from the model respectively; square, diamond and circle markers identify the (s, T_s) couple as $(1, 1)$, $(2.5, 2.5)$ and $(5, 5)$ respectively. In all the three cases the model fits the curves reasonably well and is always providing a conservative approximation: the predicted error is always larger than obtained from actual data.

Fig. 4.3(b) shows contour plots of the average approximation error. Bold lines are marked with the actual error value, which is most always smaller than 2 %, but for $1 \le T_s \le 3$ and $s < 1.5$ where it is slightly larger than 5 %. This is mainly due to two effects: the randomness of the original signal is higher and the linear fitting is less appropriate for small s and T_s as a consequence of the stronger impact of fast fading and a slower prediction reliability degradation respectively.

Finally, we conclude that Gaussian random walks can be used as a valid model for short term prediction errors since they can reproduce the main characteristics of the original random process. Also, random walks allow for an easier analysis of prediction based optimization problems: in fact, it is possible to approximate the distribution of the prediction error as a sum zero mean Gaussian variables: one of variance $B(s, T_s)$ accounting for the sequence inherent randomness and k with variance $A(s, T_s)$ each to account for decreasing reliability of the prediction after k steps. Hence, since the model is conservative with respect to the uncertainty introduced by imperfect prediction, optimization algorithms' performance obtained through this approximation are conservative as well. Thus it will be possible to derive optimization algorithms leveraging on the prediction reliability in order to mitigate the effects of uncertainties.

Part II : Resource Allocation Optimization in Mobile Networks

Summary

This second part of the thesis is dedicated to resource allocation optimization in mobile networks: here the focus is on how resources are distributed among users and time to optimize a given objective function. Thus, the solutions proposed do not have the same granularity of network schedulers (i.e. 1 ms for LTE). Instead, they make decisions on time scale of hundreds of milliseconds or even a few seconds at once. The idea, is that the decisions made here are used to modulate the procedure of the normal network schedulers that are still in charge of making the fine allocations.

In Chapter 5 we present a resource allocation scheme that minimizes the resources used to stream a multimedia content and can be executed on a mobile phone. First, we discuss the optimal solution obtained using perfect prediction and, then, we propose an iterative scheme that adapts both the prediction and the solution as soon as new information are available. In such a way, the impact of prediction uncertainty is mitigated thanks to the adaptiveness of the approach. The adaptive solution trades saved resources to avoid outage risk and the more uncertain is the prediction the more conservative is the solution.

In Chapter 6 we discuss how resource allocation can be managed in a centralized way by base stations. In this case, all users must be served with minimum outage time and maximum achievable quality. First, a convex optimization problem is written to find the optimal solution, then, a finite-complexity algorithm is proposed to approximate the solution. The results obtained on synthetic traces show that the approximation achieve performance similar to the optimal solution if allowed enough time to converge. Moreover, in many situations, a greedy one shot solution can be as good as the optimal.

In Chapter 7, the last of this part, we focus on admission control. In fact, the previous chapter shows that, if a user exists with very bad signal quality, the overall performance is severely impacted. Here, we devise a MILP formulation that accounts for guaranteed QoS by selecting the largest set of users that can be served at the same time by the base station without violating they contractual agreements. Then, an iterative approach based on a reduced linear program is given to approximate the optimal solution in a finite amount of time. The results show that both the optimal solution and its approximation can effectively guarantee the desired QoS by deciding which users to be allowed into the system.

Chapter 5

Mobile Network Resource Optimization under Imperfect Prediction

In this chapter, we propose a resource allocation algorithm for mobile networks that leverages link quality prediction and prediction reliability. Our solution exploits simple AR filters to compute short term prediction [103, 192] and the analytical data rate model presented in Chapter 3. Thus, we do not assume a perfect knowledge of future evolution of the network as in [21,22], and we are able to extend the prediction horizon from tens of seconds [39] to the order of minutes.

We develop an optimal resource allocation algorithm that assumes perfect forecast and a general prediction framework that combines short and medium/long term prediction. Subsequently, we introduce our Imperfect Capacity prediction-Aware Resource Optimization (ICARO) algorithm, which iteratively uses the optimal algorithm on a predicted data rate sequence. Finally, we validate our approach on data traces derived from measurements performed in Berlin by the MOMENTUM project [160] and we show that ICARO achieves almost optimal outage performance and outperforms solution with shorter prediction horizon.

The rest of the chapter is structured as follows: Section 5.1 provides a summary of the related work. Section 5.2 describes the system model and assumptions. In Section 5.3 we present the omniscient resource allocation algorithm. Section 5.4 analyzes future prediction feasibility and its limits: Section 5.4.1 gives details about the filtering technique used to obtain short term predictions and Section 5.4.2 discusses the statistical tools for medium to long term forecasting. Section 5.5 provides our solution for resource allocation under imperfect prediction and the performance of this algorithm is analyzed in Section 5.6.

5.1. Related work

Several recent papers, for example [4, 21, 22, 39, 52], optimize mobile network resources by exploiting future knowledge in order to save both energy and cost. The main idea is that it is better to communicate when the signal quality is good and refrain from doing so when the signal quality is bad: better signal quality results in higher spectral efficiency and fewer resources are needed to send the same amount of data.

For instance, the authors of [21] provide an optimal resource allocation algorithm exploiting perfect future knowledge, while the authors of [22, 52] provide a linear programming (LP) formulation of the resource allocation problem and solve it with optimal LP solvers. In [39], actual mobility prediction tools have been used to validate proportionally fair scheduling algorithms for cellular networks.

This paper considers a general formulation of the resource allocation problem which is not limited to video delivery. As in other works, it assumes that it is always possible to know in advance what content the user will be interested in [202] in order to be able to be able to prefetch data. Also, we relax the assumption of perfect knowledge of future system conditions, taking into consideration prediction techniques and their reliability.

Network capacity prediction has been studied, for example, in [103, 192] and [4]. These papers evaluate ARMA and GARCH filtering techniques that account for sequences of random variables that have the same (homoscedastic) or different (heteroscedastic) finite variance respectively. Other papers [35, 81] investigate mobility prediction using either Markovian estimators or trajectory-based forecasting techniques. Margolies at al. [39] propose an advanced map-based solution to extend the network forecast to tens of seconds.

A key aspect of our solution is that it accounts for the statistic model we developed in [4], which extends those proposed in [112, 197] to account for imprecise information. This approach allows us to extend the prediction horizon to the order of minutes, without requiring very complex computations.

5.2. System model

In this paper we address the downlink from a base station of a mobile network (eNodeB) to a single receiver (UE). To simplify the description of the problem, we consider slotted time with slot duration t and thus the quantities discussed in the paper are discrete time series. We use i, j, and k to refer to slot indices. The quantities of interest are:

- Position $P = \{p_i \in [0, P_{\max}],\ i \in \mathbb{N}\}$, where p_i is the distance between UE and eNodeB and $P_{\max}$ is the coverage range.

- Active users $N = \{n_i, i \in \mathbb{N}\}$, where n_i is the number of active users that are in the same cell as the UE. It reflects the congestion level of the cell in slot i.

- Signal to interference plus noise ratio (SINR) $S = \{s_i \in \mathbb{R},\ i \in \mathbb{N}\}$, where s_i is obtained from p_i as follows:

$$s_i = s_0 p_i^{-\alpha} f_F. \tag{5.1}$$

Here, s_0 is a system constant, α is the path loss exponent and f_F is a random multiplicative term to account for fast fading.

- User cell capacity $C = \{c_i \in [0, C_{\max}],\ i \in \mathbb{N}\}$, where c_i represents the average capacity obtained by the user during slot i. $C_{\max}$ is the maximum capacity allocable to the UE, given the specific mobile technology. We compute c_i as a function of s_i and n_i through

$$c_i = c_0 g_c(s_i, n_i), \tag{5.2}$$

where c_0 is a system constant and g_c is a technology dependent function which models system level variables such scheduling policy, congestion, spectral efficiency, etc. In the rest of the paper we consider LTE as the mobile network technology and we adopt the model in [197], which provides a closed form expression for f_F and g_c for a user at a given distance from the base station, when another $n-1$ users are uniformly distributed in the cell area and proportionally fair scheduling is used.

- Receive rate $R = \{r_i \in [0, c_i],\ i \in \mathbb{N}\}$: this is the rate at which the base station sends data to the UE in slot i.

- Download requirement $D = \{d_i \in [0, D_{\max}], i \in \mathbb{N}\}$, where $D_{\max}$ is the maximum data consumption rate. In slot i, the user consumes d_i bytes of data if they are available. If at any time the user receives more data than required, the excess can be stored in a buffer for later use.

- Buffer state $B = \{b_i \in [0, B_M],\ i \in \mathbb{N}\}$, where b_i is the buffer level and B_M is the buffer size in bytes.

- Buffer under-run time $U = \{u_i \in [0, 1],\ i \in \mathbb{N}\}$ is the fraction of slot i for which no data was available to satisfy the download requirements.

The aforementioned quantities are linked as follows:

$$b_{i+1} = \min\{\max\{b_i + r_i - d_i, 0\}, B_M\} \tag{5.3}$$

$$u_i = \begin{cases} \max\{d_i - r_i - b_i, 0\}/d_i & d_i > 0 \\ 0 & d_i = 0 \end{cases}. \tag{5.4}$$

The buffer fills (up to the full buffer B_M) whenever the download rate is higher than the consumption rate, $r_i > d_i$. In case $r_i < d_i$, the algorithm empties the buffer and accumulates buffer under-run time whenever $b_i + r_i < d_i$. In what follows, we refer to function $y = g_y(x)$ as g_y. Similarly, we refer to the probability density function and the cumulative density function (CDF) of a random variable X as $f_X(x)$ and $F_X(x) = \int_{-\infty}^{x} f_X(y)\mathrm{d}y$ and with μ_X and σ_X to its mean and standard deviation.

5.3. Resource allocation optimization with perfect forecast

The resource allocation problem aims at finding the optimal rate time series R that satisfies the download requirements D by using the available capacity C in the most efficient way. We define the following objective function:

$$O = \{o_i = r_i/c_i \in [0,1],\ i \in \mathbb{N}\}, \tag{5.5}$$

where o_i is the fraction of the available capacity used in slot i and represents a cost. Note that the same rate r has a different cost $o_i > o_j$ if the available capacity $c_i < c_j$. We obtain the following optimization problem:

$$\begin{aligned} \underset{R}{\text{minimize}} \quad & \sum_i o_i \\ \text{subject to:} \quad & \sum_i u_i = \sum_i u_i^*, \\ & b_i \leq B_M,\ \forall i \in \mathbb{N}, \end{aligned} \tag{5.6}$$

where $\sum_i u_i^*$ is the minimum feasible buffer under-run time. To minimize this cost function, the base station should send more data when the available capacity is high and use just the minimum rate required to avoid a buffer under-run when the capacity is low.

The solution of Eq. 5.6 is the optimal resource allocation strategy R^* that achieves the minimum buffer under-run time $\sum_i u_i^*$ at the lowest cost $\sum_i o_i^*$. If the sequence C is known a priori, various offline algorithms can be used to determine the optimal resource allocation. We propose a simple algorithm that we call Split & Sort (S&S), which splits the optimization horizon into windows so that allocation decisions belonging to two different windows can be made independently. Within each window slots are used in descending order of predicted capacity. The last slot of each window is called a break-point.

S&S computes the optimal solution of Eq. 5.6 by using the following rules: *i*) define the break-point e_l as the last slot for which all previous rates are finalized (i.e., no more rate can be used in slots up to e_l) which requires that either $b_{e_l} = B_M$ or $r_k = c_k, \forall e_{l-1} < k \leq e_l$; *ii*) define the optimization window $[e_l + 1, m]$, where e_l is the last break-point slot and the rate allocated in all slots in $e_{l-1} < k \leq e_l$ is finalized; *iii*) starting from $l = 0$, $e_l = 0$ and $m = 1$ the algorithm accounts for the slots in the set $\{e_l + 1, \ldots, e_l + m\}$ to satisfy the requirements up to slot $e_l + m$; the algorithm chooses a slot if it has the highest capacity among the unused ones in the set. *iv*) the algorithm either increments l, updates e_l and resets $m = 0$ if a break-point is found or increments m. The complete Split & Sort algorithm is given in Algorithm 1. $\text{sAdd}(X, x)$ adds the element x to the sorted list X in the correct position, $\pi(c_i)$ gives the position in C of the element c_i and $\text{shift}(D, u_j, j)$ is a shift function that recomputes the requirements sequence D accounting for a buffer under-run event u_j in slot j. The following conditions are used:

- $\text{I}_1 := \exists\ e_l < j \leq e_l + m \mid b_j = B_M$ to verify whether a full buffer state is reached,

- $\mathrm{I}_2 := \sum_{j=e_l+1}^{e_l+m} c_j - r_j = 0$ to verify whether all of the available capacity is used, and

- $\mathrm{I}_3 := \sum_{j=e_l+1}^{e_l+m} r_j - d_j = 0$ to verify whether all of the download requirements have been satisfied.

Algorithm 1 Split & Sort Algorithm (S&S)

```
Input: the knowledge of the future capacity availability $C$, the future download requirements $D$
  and the initial buffer level $B_0$.
Output: $R = \mathrm{SS}(C, D, B_0)$
  $l = 0, e_l = 0$ // set the starting point
  $b_{e_l} = B_0$ // set the starting buffer
  $r_{e_l} = 0$, $R = \emptyset$ // set the starting allocation
  while $|R| < |D|$ do
    $m = 1$ // set the initial window size
    $S = \emptyset$ // sorted capacity vector initialization
    while $\neg\mathrm{I}_1 \wedge \neg\mathrm{I}_2 \wedge |R| + m < |D|$ do
      $S = \mathrm{sAdd}(S, c_{e_l+m})$ // add an element to the sorted capacity list
      $i = 1$
      while $i \leq m \wedge \neg\mathrm{I}_3$ do
        $r_{\pi(s_i),\mathrm{old}} = r_{\pi(s_i)}$ // store previous allocation
        $r_{\pi(s_i)} = \min\{r_{\pi(s_i)} + d_{e_l+m}, c_{h_i}, B_M - b_{\pi(s_i)}\}$
        $b_{\pi(s_i)+j} = b_{\pi(s_i)+j} + r_{\pi(s_i)} - r_{\pi(s_i),\mathrm{old}}, \forall\, 1 \leq j \leq m - \pi(s_i)$
        $i = i + 1$
      end while
      $m = m + 1$ // update the window size
    end while
    if $\mathrm{I}_1$ then
      $l = l + 1, e_l = j$ // new break-point
    else
      $l = l + 1, e_l = e_{l-1} + m$ // new break-point
      if $\mathrm{I}_2$ then
        $u_j = \max\{d_j - r_j - b_j, 0\}/d_j$
        $D = \mathrm{shift}(D, u_j, j)$ // shift of requirements
      end if
    end if
    $R = \{R, r_{e_{l-1}}, \ldots, r_{e_l}\}$ // update the allocation
  end while
  return $R$
```

In the following we prove the optimality of Algorithm 1 and discuss the behavior of the algorithm when knowledge of the future capacity is not perfect.

Theorem 1 (Split & Sort Optimality). *If R is a solution of Algorithm 1 with C and D as inputs and it achieves a buffer under-run time $\sum_i u_i$ and cost $\sum_i o_i$, then there exists no other allocation strategy $R' \neq R$ for C and D that obtains performance $\sum_i u'_i$ and $\sum_i o'_i$, for which ($\sum_i u'_i <$*

$\sum_i u_i) \vee (\sum_i u' = \sum_i u_i \wedge \sum_i < \sum_i o_i)$, *i.e., it has either a lower buffer under-run time or the same buffer under-run time and a lower cost.*

In the following we will prove the theorem by contradiction: we show that is impossible that a solution exists which is both different from that provided by S&S and achieves better performance, due to either the stopping conditions of the algorithm or the ordering of the decisions within an optimization window.

Proof: Theorem 1 can be proven by contradiction on the following hypotheses:
Assume a solution $R' \neq R$ exists so that

1. either $\sum_i u'_i < \sum_i u_i$ (shorter buffer under-run time)

2. or $\sum_i u'_i = \sum_i u_i \Rightarrow \sum_i o'_i < \sum_i o_i$ (cheaper)

For 1) R cannot satisfy the requirements D in all the slots, thus $\sum_i u'_i < \sum_i u_i \Rightarrow \exists\ j$ s.t. $r_j + b_j < r'_j + b'_j < d_j$. Since $R' \neq R$, they must differ before or on slot j in order to cause the larger under-run time, because any variation later than that cannot decrease $\sum_i u'_i$. Since R is obtained using Algorithm 1 and must result in $u_j > 0$, then for all the slots belonging to the analysis window $[e_{l-1} + 1, e_l]$, where $e_l = j$ the whole available capacity must have been used, which means R' cannot use more capacity there to avoid the buffer under-run. R' cannot use more capacity before slot e_{l-1} either, since that would impact a window already completed (ended because of condition I_1). Thus, $\sum_i u'_i \geq \sum_i u_i$ if the two strategies are different, which contradicts the first hypothesis.

For 2) it is $(R \neq R') \wedge (\sum_i u_i = \sum_i u'_i) \wedge (\sum_i o_i < \sum_i o'_i)$, thus the two strategies must differ in at least two slots j, k, where $c_j > c_k$ and $(r_j < r'_j) \wedge (r_k > r'_k)$. The two slots j, k cannot belong to the same window, because Algorithm 1 uses the slots from a sorted list and finishes either with a full buffer or when the whole capacity has been used. The two slots j, k cannot belong to different windows either, because if $j < k$, it would have been possible to use more capacity earlier in the allocation which is not possible due to the stopping conditions of the algorithms, whereas if $j > k$, a cheaper slot later in the sequence could have been used instead of a more expensive one earlier in the sequence. However, this is not possible due to either the fact that the more expensive slot must have been used in order not increase $\sum_i u_i$ (stopping condition I_2) or because of the ordered selection of the slots (stopping conditions I_1 or I_3). Thus, $\sum_i o'_i \geq \sum_i o_i$ if the two strategies are different, which contradicts the second hypothesis.

Thus, assuming that an allocation strategy R' provides a better solution than that obtained using Algorithm 1 violates the hypotheses of the theorem, which is therefore proved. ■

Algorithm 1 will be later used in Section 5.5 in an iterative procedure to compute the resource allocation when the knowledge of future capacity is inaccurate.

5.4. General forecast model

In this section we propose a general model describing the forecasting reliability of a system. In particular, we split our model in three time periods based on the prediction horizon:

The **short term** period considers the near future and predicts capacity through time-series filtering techniques [103, 192]. It is characterized by the reliability time τ_p, which defines how many slots of the sequence can be predicted. We discuss this in Section 5.4.1.

The **medium term** period describes the evolution of the system in terms of available capacity statistics. During this period one or more network cells can be accounted to in the mobility predictor: Markovian predictors [81] can usually compute the likelihood of visiting a given cell, while trajectory-based predictors [35] provide a more accurate estimate by computing the actual distribution of the user position over time.

The **long term** period provides an overall statistical evaluation of the available capacity availability based on the steady state distribution of the user position in the network. Both the medium and the long term periods are discussed in Section 5.4.2.

5.4.1. Short term forecast with filters

This section addresses the reliability time τ_p achievable by filtering techniques applied to available capacity time series. In particular, we study autoregressive-moving average (ARMA) filters and their setup according to the system dynamics defined by the slot time t and the user speed v. We opted for ARMA instead of GARCH [103], since capacity elements belonging to the short term period are characterized by the same finite variance.

For each $(t \in [0.5, 5], v \in [0.5, 5])$ tuple we consider a set of 100 capacity traces computed using Eqns. (5.1) and (5.2) as per [197], starting from the mobility paths of a user moving at constant speed in a random network deployment. We apply the Box-Jenkins [200] method to determine the type and the order of the filter to be used with each sequence. Through the analysis of autocorrelation and partial autocorrelation plots, we find that the best technique for our sequences consists of simple autoregressive (AR) filters of order τ_F, and that τ_F is inversely proportional to the tv product.

Subsequently, for each of the sequences we estimate filter coefficients by means of the linear least squares procedure [203] and we use the obtained filter to forecast the values of the other sequences with the same (t, v) parameters. We refer to a forecast sequence as $\tilde{C} = \{\tilde{c}_i \in [0, C_{\max}],\ i \in \mathbb{N}\}$, obtained from C and to the corresponding error $\Delta = \{\delta_i = \tilde{c}_i - c_i \in [-C_{\max}, C_{\max}],\ i \in \mathbb{N}\}$. We consider a prediction to be reliable as long as the standard deviation of the error is lower than that of the capacity, $\sigma_\Delta = \sigma_C$.

Thus, we compute μ_Δ and σ_Δ as the average and the standard deviation of all the error sequences with the same (t, v) parameters. Fig. 5.1 shows on the abscissa the prediction time index normalized on t and on the ordinate σ_Δ/σ_C the standard deviation of the prediction normalized on the standard deviation σ_C of the original series C.

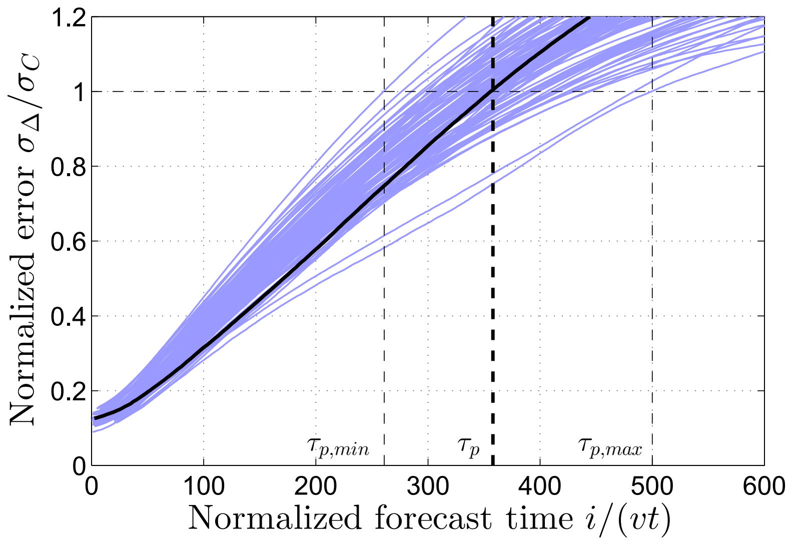


Figure 5.1: The shaded area represents how the standard deviation of the short term prediction error increases with increasing prediction distance varying the user speed v and the slot time t. τ_p represents the time after which $\sigma_\Delta \geq \sigma_C$.

While the actual steepness of the curves varies with the parameters, for all of them the normalized error standard deviation σ_Δ/σ_C approaches 1 almost linearly. Hence, we set $\tau_p = \text{argmin}_i \text{ s.t. } \sigma_{\Delta_i}/\sigma_C > 1$. In addition, we observe that both τ_p and the filter order can be approximated with simple linear models with the inverse of the tv product and that τ_p is usually 10 times as large as the order of the AR filter.

Finally, it is sufficient to tune a set filters for varying t and v and select the one to use according to the actual user mobility. Also, since filters can be normalized on σ_C it is not needed to have different filters for different numbers of active users in the cell, but it is sufficient to rescale the constant and the variance parameters of the filter.

5.4.2. Statistical models and uncertainties

For medium and long term prediction we base the model of distribution of per user capacity we started on [197], since to the best of our knowledge it is the only one which takes into account the scheduler impact and thus is able to model user contention.

To account for the impact of uncertainties on the user position and/or the number of active users in the cell we modify the expression of the capacity distribution $f_C(x)$ obtained for a specific position p_i and number of users n_i to the actual distribution of the user position $f_P(x)$ and the probability mass function $f_N(n)$ of the number of active users in the cell, as follows:

$$f_C(x) = \sum_{i\in\mathcal{N}} f_N(i) \int_0^\infty f_{F,P|i}(g_C^{-1}(x,p), p|i) \left| \frac{\partial g_C^{-1}(x,p)}{\partial x} \right| \mathrm{d}p, \tag{5.7}$$

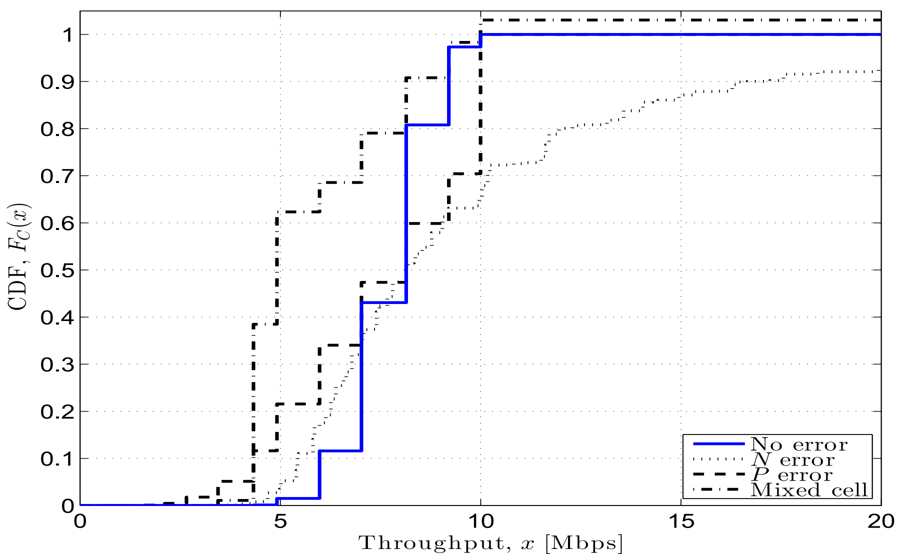


Figure 5.2: Examples of the impact of uncertainties on the capacity distribution.

where $f_{F,P}$ is the joint distribution of fading and position, g_C is the function linking the per user capacity to p and n and $\mathcal{N}$ is the support of $f_N(n)$. Since fading and user position are statistically independent, their joint distribution $f_{F,P}(x,y) = f_F(x)f_P(y)$ is the product of their distributions. Eq. (5.7) modifies the original capacity distribution weighting it through the active user probability mass function $f_N(i)$ and the user position probability $f_P(y)$; the partial derivative normalizes the integrand.

For what concerns our analysis, it is sufficient to be able to compute the per user capacity distribution by accounting for limited knowledge of the user position and traffic in the cell by means of their respective distributions.

So far, our model describes capacity only for the case when the cell the user is connected to is known perfectly. To account for different cells, it is sufficient to consider the weighted sum of the capacity distributions of single cells,

$$f_C(x) = \sum_{i \in \mathcal{C}} \rho_i f_{C,i}(x), \tag{5.8}$$

where $\mathcal{C}$ is the set of cells that can be visited in the next time period with some probability, $f_{C,i}(x)$ is the capacity distribution related to cell i and ρ_i is the probability of visiting cell i in the next time period.

Fig. 5.2 provides a few examples of the CDF obtained using the model. The solid line is representative of the capacity CDF $F_C(x)$ when both the active user number $n = 5$ and the user position $p = 500$ meters are exactly known so that the distribution is equal to the fading distribution. The dotted line accounts for an error in the number of active users in the cell so that $f_N(x) = \{0.2, 0.6, 0.2\}$ for $x = \{4, 5, 6\}$ respectively.

Conversely the dashed line is obtained by accounting for an error in the user position which has a normal distribution with parameters $\mu_P = 500$ meters and $\sigma_P = 100$ meters. Finally, the dash-dotted line is obtained by mixing together two cells with 5 and 10 users with 20% and 80% of visiting probability respectively. The piecewise-constant shape of the curves is due to the discrete relationship between SINR and bitrates.

While practical implementations of this solution can use different methodologies, in our evaluation campaign we proceed as follows. From the measurements of the MOMENTUM project and the channel model in [197] we derive the model for the capacity distribution for each cell of the network (cells are defined so that each point of the area is associated to the base station which has the strongest average SNR).

We assume the user position statistic $f_P(x)$ to be uniform in the area (i.e., we do not leverage any auxiliary information such as street topology). Similarly, we computed the cell traversal time $\tau^{(i)}$ as the ratio between the average cell width and the average user speed. Thus, we are able to define the statistical model for each cell of the network, while, for the long term period we use Eq. (5.8) over the whole area and assume a uniform distribution among cells.

5.5. Resource allocation optimization under uncertainties

The objective of this section is leveraging the concepts of the previous ones to design a network resource allocation algorithm which takes into account imperfect forecast, called Imperfect Capacity prediction-Aware Resource Optimization (ICARO). ICARO aims at minimizing the communication cost while avoiding buffer under-runs.

In particular, we use Algorithm 1 (S&S) of Section 5.3 in an iterative way. At each iteration, Algorithm 1 makes a single decisions about which rate r to use by exploiting both the AR predictor described in Section 5.4.1 and the statistical models designed in Section 5.4.2. Before describing the new algorithm, we describe how to obtain a single general capacity prediction to use with Algorithm 1. In order to account for the three time periods described in Section 5.4 we proceed as follows:

1) The short term prediction $\tilde{c}_i^{(F)}$ with $i \in [0, \tau_p]$ is obtained from the known past capacity information [12] and choosing the filter order τ_F and coefficients based on the user speed v.

2) The medium term model $f_{C,i}(x)$ is computed as the superposition of the cells $j \in \mathcal{C}$ that the user is likely to visit in the i-th time period, each of them accounted for according to their user position $f_{P,j}(y)$ and active user number $f_{N,j}(z)$ statistics by Eq. (5.8). Similarly, the duration of the i-th time period $\tau_i - \tau_{i-1}$, is obtained as a weighted sum of cells traversal time $\tau^{(j)}$ related to cell $j \in (C)$.

3) During the i-th time period $D_i = \sum_{j=\tau_{i-1}}^{\tau_i} d_j$ bytes have to be downloaded to avoid a buffer under-run. The maximum cell efficiency is achieved when only the slots with the highest capacity are used.

4) The highest threshold $c_{T,i}$ is computed so that the average amount of data obtained by selecting only the slots with larger capacity than c_T is larger than $D_i/(\tau_i - \tau_{i-1})$:

$$c_{T,i} = \max_y \ \text{ s.t. } \int_y^{\infty} x f_{C,i}(x)\mathrm{d}x \geq D_i/(\tau_i - \tau_{i-1}). \tag{5.9}$$

5) The i-th time period is modeled as a sequence of $\tau_i - \tau_{i-1}$ values

$$\tilde{c}_j^{(M,i)} = \begin{cases} c_{T,i} & j > (1 - F_{C,i}(c_{T,i}))(\tau_i - \tau_{i-1}) \\ 0 & \text{otherwise} \end{cases}, \tag{5.10}$$

where $F_{C,i}(c_{T,i})$ is the probability of the capacity being lower than $c_{T,i}$, thus $(1 - F_{C,i}(c_{T,i}))(\tau_i - \tau_{i-1})$ is the average number of slots with larger capacity than the threshold.

6) Steps 2 to 5 are repeated and new time periods are added in the sequence if their reliability is sufficient. In our evaluation campaign we consider two periods only, each related exactly to one cell: the current and the following which we choose according to the current mobility direction.

7) Compute τ_o as the offset time when the user first entered in the cell.

8) Obtain the predicted capacity sequence as the concatenation of the previously computed time period sequences:

$$\tilde{c}_i = \begin{cases} c_0 & i = 0 \\ \tilde{c}_i^{(F)} & 0 < i \leq \tau_p \\ \tilde{c}_i^{(M,1)} & \tau_p < i \leq \tau_1 \wedge \tau_1 > \tau_p + \tau_o \\ \tilde{c}_i^{(M,2)} & \max(\tau_o + \tau_p, \tau_1) < i \leq \tau_2 \\ \dots & \\ \tilde{c}_i^{(M,n)} & \tau_{n-1} < i \leq \tau_n \end{cases}, \tag{5.11}$$

where τ_n is the duration of the whole sequence, c_0 is the known present capacity, and $\tilde{c}_i^{(M,1)}$ is the current period capacity distribution. $\tilde{c}_i^{(M,1)}$ is modified by accounting for the time passed from when the user first entered the cell in τ_o: for each passed time slot one sample is removed either from the beginning if $c_0 < cT, 1$ (higher capacity can be found later, since the current capacity is lower than the current capacity threshold) or from the end otherwise (capacity is sufficiently high).

Fig. 5.3 shows an example of a mixed model sequence: the upper part compares the ground truth (C as a thin solid line) to the short term ($\tilde{C}^{(F)}$ as a thick solid line) and the medium-long term ($\tilde{C}^{(M,1)}$ and $\tilde{C}^{(M,2)}$ as dashed line) predictions respectively. The lower part is a map of the user movement (central horizontal arrow) and the coverage areas of different cells (dashed circles). The shaded area highlights the uncertainties in future user position. Dash-dotted lines crossing the figures mark τ_p, τ_1 and τ_2 instants.

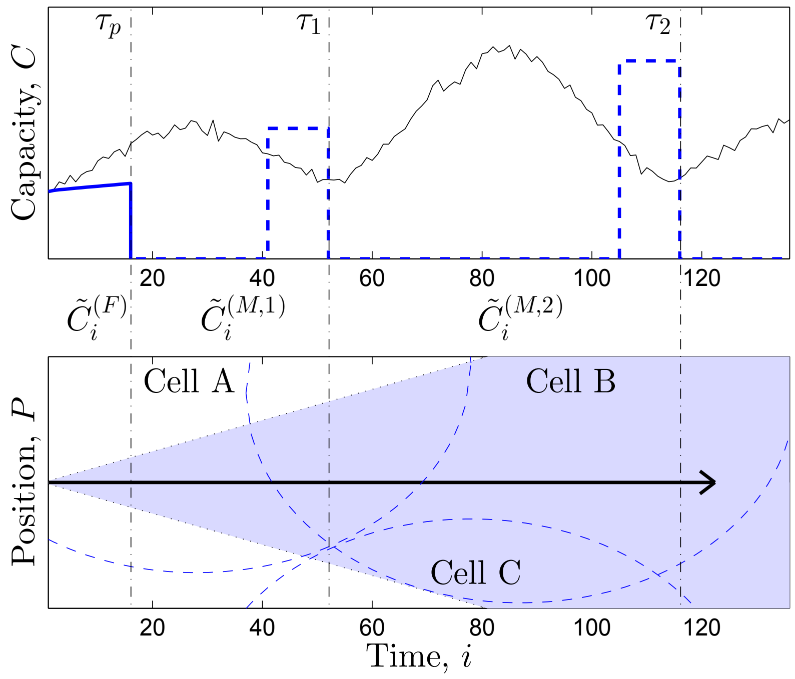


Figure 5.3: Example of the general prediction model and related user position.

In every time slot, ICARO (Algorithm 2) computes the mixed forecast sequence of Eq. (5.11) and uses Algorithm 1 (S&S) to allocate the rate of the current slot. The algorithm iterates until the requirements are completely satisfied.

Algorithm 2 Imperfect Capacity prediction-Aware Resource Optimization (ICARO)

```
Input: the future download requirement $D$, user speed $v$ and position $p$, $\tau_F$ past values of the capacity sampled with $t$ period, the capacity statistics $f_{C,i}(x)$ and time period traversal time $\tau_i$ for the next predictable time periods.
Output: $R, O, U$
  $s = 0$ // set the starting point
  $b_s = B_0$ // set the starting buffer
  $r_s = 0$, $R = \emptyset$ // set the starting allocation
  while $\sum_{i=s}^{|D|} d_i \geq b_s$ do
      compute $\tilde{C}$ as per Eq. (5.11)
      run $\hat{R} = \mathrm{SS}(\tilde{C}, D, b_s)$ // allocation is computed using Algorithm 1 on the predicted sequence of Eq. 5.11
      $r_s = \min(\hat{r}_1, c_s, B_M - b_s)$ // rate to be used
      compute next buffer state $b_{s+1}$ according to Eq. (5.3)
      compute buffer under-run $u_s$ according to Eq. (5.4)
      compute cost $o_s$ according to Eq. (5.5)
      $s = s + 1$
      $D = \{d_i, s < i \leq |D|\}$ // remove the first element from the requirements sequence
  end while
  return $R, O, U$
```

The rationale for using the S&S algorithm on the mixed forecast sequence is that its operational principle, that selects which slot to use in descending order, still works under uncertainties and provides a solution which is conservative (as the highest capacity slots are placed last) to avoid under-runs, and aggressive (as the allocation priority is given to the most reliable slots) to optimize allocation costs. In the following, we provide a few examples of the algorithm:

Ordering the short term forecast: the elements of the short term prediction sequence can be assumed to have the same order of those of the actual sequence. In fact, as we showed in Section 5.4.1, $\sigma_{\delta,i}$ is increasing with i, thus if $\tilde{c}_i^{(F)} > \tilde{c}_j^{(F)}$ and $j > i$, then the probability of having the same ordering is larger than that of opposite order ($\mathrm{P}[c_i > c_j] > \mathrm{P}[c_i \leq c_j]$). Thus, the S&S algorithm can be used on the short term prediction, because its order is likely to match that of the actual sequence.

Comparing short and medium term forecast: the i-th medium term period is represented as a sequence of $(\tau_i - \tau_{i-1})F_{C,i}(c_{T,i})$ zero capacity slots while the remaining slots are equal to $c_{T,i}$, which represent a worst case scenario computed on the known capacity distribution. Hence, if the short term prediction is lower than $c_{T,1}$ only the minimum rate is used, since from the statistical model slots of higher capacity are expected to come later. Conversely, if the short term prediction is larger than $c_{T,1}$, then it is more likely that the remaining slots will be lower than the threshold (see also step 8 of the sequence creation). In other words, running the S&S algorithm on this sequence ensures that it buffers enough data to avoid using the zero-capacity slots by exploiting those with a capacity larger than $c_{T,1}$.

Buffering: the algorithm will always try to use the slots above threshold in each time period and bridge the gaps between those by using the buffer. By positioning the slots with highest capacity at the end of each time periods we ensure that the algorithm is conservative. Finally, the maximum buffer size B_M limits the optimization horizon of the algorithm: in fact, the maximum time that the system can last without using any capacity is given by $B_M/(\sum_i d_i/|D|)$. Hence, the buffer size has a significant impact on the algorithm's performance which we analyze in the next section.

Fig. 5.4 shows an example of ICARO's performance compared to the optimal boundary (OPT) obtained with perfect forecast and to the trivial (FULL) solution which maintains the buffer as full as possible at all times. The top three plots show the used rate R of the three algorithms: ICARO, OPT and FULL from the top. The shaded areas represent the used part of the total available capacity (solid line). While FULL continues to fill the buffer during the low quality period ($i = 25$), OPT just uses the needed quantity to harness the best part of the second cell ($i = 50$). ICARO's decisions, even though slightly more conservative (i.e.: $i = 80$), are very similar to OPT's. The last two plots show the buffer and the cumulative cost variation respectively.

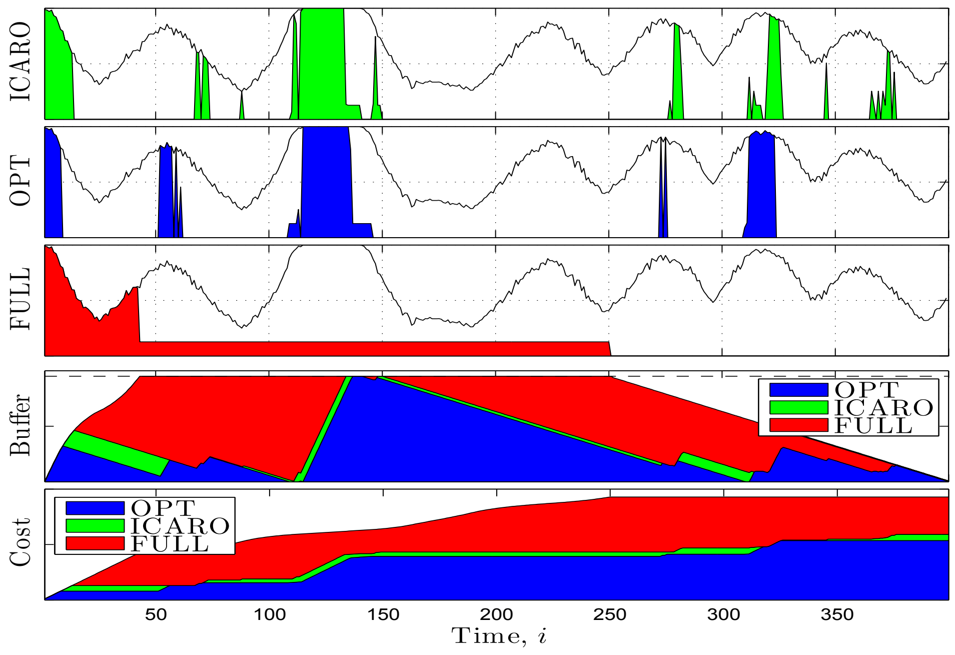


Figure 5.4: Comparison among the three main algorithms.

5.6. Results

In this section we provide an analysis of the performance of our algorithm. In particular we compare ICARO against the following algorithms:

• OPT: the optimum offline allocation computed with the optimal S&S algorithm on the exact capacity time series.

• FULL: the most conservative approach which just fills up the buffer as soon as possible and maintains it as full as possible until the download requirements are satisfied.

• OPT(x): an optimal algorithm that iteratively makes the decision on the current slot by running the S&S algorithm on the first x samples of the exact future capacity. This algorithm targets a full buffer (or the sum of the remaining requirements if it is lower than the buffer size) at the end of the optimization window. This algorithm represent the performance upper bound for any solution using at most x samples of prediction.

Our main performance metrics are the objective function O and the buffer under-run time U. To compare the results of every tested configuration, we adopt the average cost $\xi = \sum_i o_i/|O|$, the average cost saving $\eta = (\sum_i o_{i,\text{FULL}} - o_{i,\text{ICARO}})/\sum_i o_{i,\text{FULL}}$ obtained by our algorithm, and the average buffer under-run time increase $\zeta = \sum_i u_{i,\text{ICARO}} - \sum_i u_{i,\text{OPT}}$. In addition, we study the impact of the parameter x on OPT(x) and compare it to our solution.

Our evaluation campaign considers an LTE network scenario based on the pathloss data provided by the MOMENTUM project [160] and accounts for both vehicular and pedestrian mobility ($\mu_v = 5$ and 1 meters per second respectively). For each evaluation round we generate a random

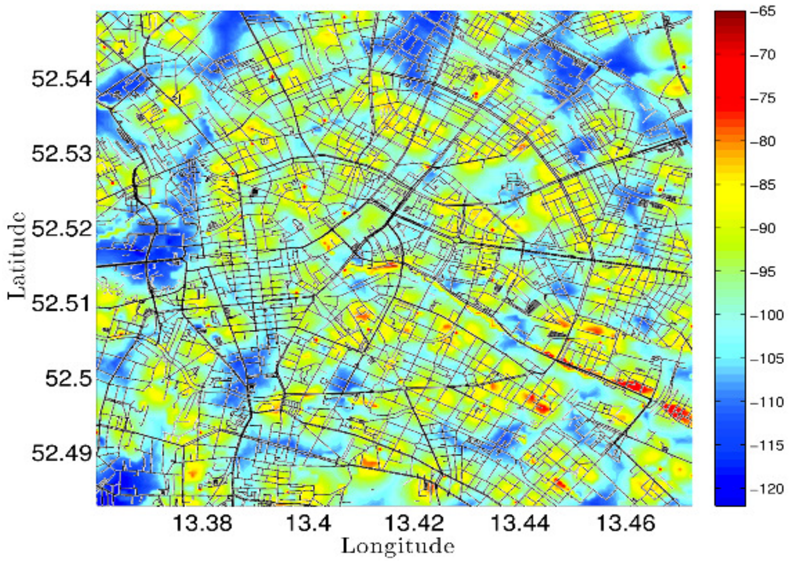


Figure 5.5: Pathloss map of Berlin as measured by the MOMENTUM project.

mobility trace in a 12×6 square kilometer area of Berlin (centered at latitude $52.52°$ North and longitude $13.42°$ East. From the mobility trace, we generate a pathloss trace computed on the pathloss map of Fig. 5.5. Finally, we account for fast fading as per in the analysis of proportionally fair scheduling in [197] to obtain the capacity trace. In all experiments we consider an average number of active users $N = 5$ uniformly distributed.

Each of the trace represents the ground truth for one of our experiments and consists of 4000 capacity samples. From the whole sequence we estimate the parameters of the AR filters that we use for ICARO (see Section 5.4.1 above), while we run the four algorithms on 10% of the samples only, chosen starting from a random starting point of the trace. In order to provide ICARO with the medium and long term parts of the prediction we assume the following:

- a user stays in a cell for an average time equal to the ratio between the average cell width (that we computed numerically for each cell from the MOMENTUM data) and the user speed;
- the capacity distribution of a given cell is computed as per Section 5.4.2 assuming the position to be uniformly distributed in the area of the cell;
- the current and the next cells are known;
- the long term distribution is the combination of all the cells visited during the whole trace.

Fig. 5.6 and Fig. 5.7 show the main results of our evaluation campaign for pedestrian and vehicular mobility respectively: in both cases we vary the requirement over capacity ratio $(\sum_i d_i / \sum_i c_i) \in [0.1, 0.9]$, and the normalized buffer size $(B_M \sum_i c_i / \sum_i d_i) \in [1, 200]$.

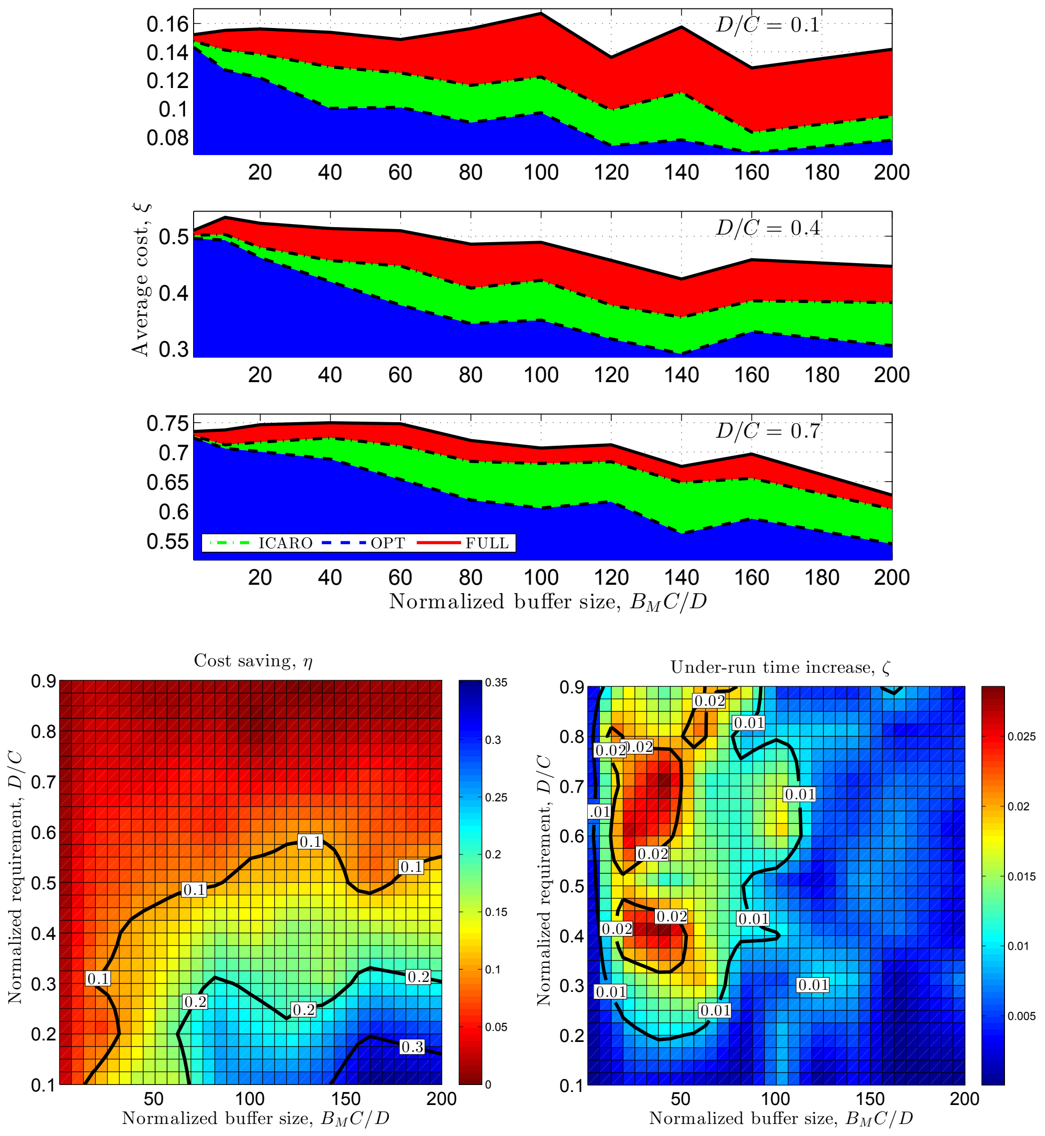


Figure 5.6: Performance comparison among ICARO, OPT and FULL with vehicular mobility. ξ, η and ζ are plotted on the left, center and right respectively.

Fig. 5.6 (left) shows the average cost ξ of the three main algorithms (OPT, ICARO and FULL as solid, dashed and dash dotted lines, respectively) varying the buffer size (x-axis) for $\sum_i d_i / \sum_i c_i = \{0.1, 0.4, 0.7\}$ (upper, center and lower plots). The variable horizon algorithm OPT(x) is accounted for in Fig. 5.8 and Fig. 5.9 for better readability.

In the upper plot the download requirements are moderate and both OPT and ICARO are able to obtain a normalized cost lower than 0.08 (corresponding to 80% of the $\sum_i d_i / \sum_i c_i$), while FULL often needs more than 100% of the average requirements ($\xi \geq 0.1$). The performance is similar in the other plots and ICARO is always better than FULL and close to OPT. As expected, ICARO performance improves when the buffer is larger and the requirements are lower. Notably,

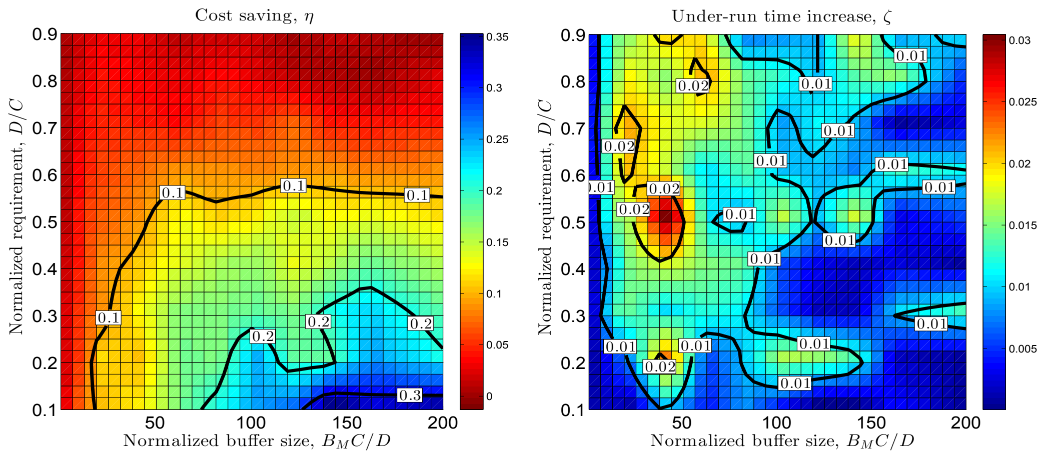


Figure 5.7: Performance comparison with pedestrian mobility.

when the buffer is very small the three algorithms perform almost exactly the same as a too small buffer does not allow leveraging forecast information.

The central figure shows contour plots of ICARO's efficiency η using $B_M \sum_i c_i / \sum_i d_i$ as abscissa and $\sum_i d_i / \sum_i c_i$ as ordinate: the curves are labeled according to the cost savings achieved and the area color changes to red where the savings are lower than 10%, while it changes to cyan and blue when it is higher than 15%. Again the best results are obtained for medium-large buffer size and small requirements where ICARO is about $25 - 30\%$ cheaper than FULL. On average, ICARO is 10% worse than OPT.

The figure on the right shows how close ICARO is to the optimal buffer under-run time obtained by both OPT and FULL. We plot ζ using the same coordinates as those of the previous figure. Here the blue part of figure highlights where ICARO is able to achieve almost optimal performance ($\zeta \leq 0.01$), while green and red areas correspond to slightly worse performance ($0.01 < \zeta < 0.04$). Notably, for no parameters the buffer under-run time was larger than 0.05, and the worst performance is obtained for low buffer sizes.

Fig. 5.7 provides results equivalent to those of the previous set of figures, but obtained for vehicular mobility. Here, all the trends identified above are confirmed and ICARO performs slightly worse than for pedestrian mobility. This is chiefly due to the higher variability of the capacity traces.

Since ICARO gives priority to avoiding buffer under-runs, it can obtain higher cost savings when the ratio between requirements and available capacity is lower. Thus, since ζ is always lower than 0.05, the algorithm is able to effectively trade off cost efficiency for robustness and it is able to achieve up to 30% cost reduction when the conditions are favorable, but it never behaves too aggressively when the future capacity estimation does not allow to do so.

Fig. 5.8 compares the results of the last algorithm OPT(x) against the length of the prediction horizon x for both vehicular (left) and pedestrian mobility (right). We plot the results of

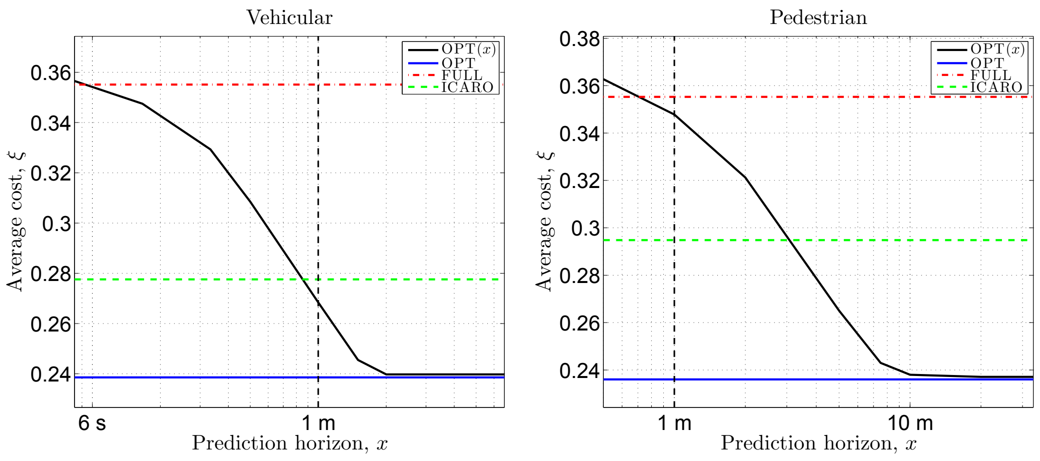


Figure 5.8: OPT(x) performance against prediction horizon x compared with the other algorithms for vehicular (left) and pedestrian mobility (right).

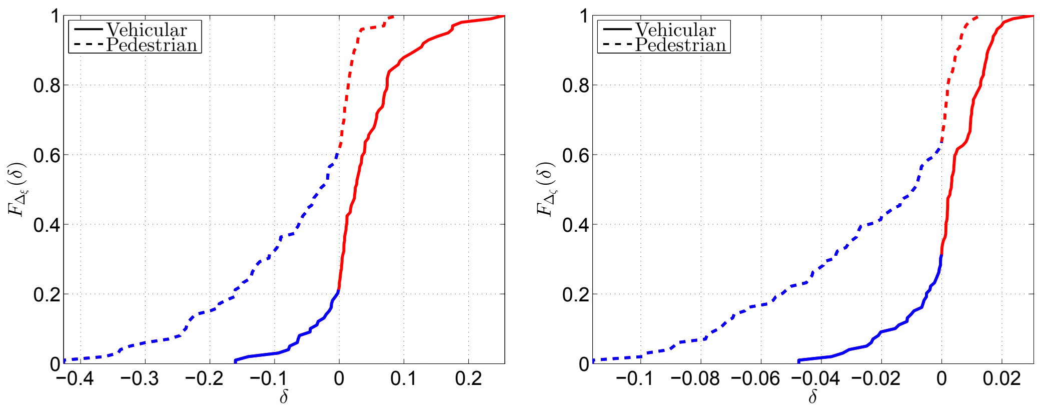


Figure 5.9: Performance gap CDF between ICARO and OPT(x) for $x = 60$ seconds. Cost and buffer under-run time gaps on the left and right respectively.

simulations run for $\sum_i d_i / \sum_i c_i = 0.3$ and $(B_M \sum_i c_i / \sum_i d_i) = 100$. OPT$(x)$ is plotted as a solid black line and clearly shows that performance improves with increasing x, starting from about the value achieved by FULL (red dash-dotted line) and reaching the OPT (blue solid line) performance at about 2 and 10 minutes prediction horizon for vehicular and pedestrian mobility.

In addition, we plot ICARO performance (green dashed line) and one vertical line to mark the 1 minute horizon, which is often used (e.g.: [39]). ICARO is performs very close to the optimal algorithm with 1 minute horizon for vehicular mobility and outperforms it for pedestrian.

Fig. 5.9 plots the CDFs of $\Delta_\xi = \xi_{\text{ICARO}} - \xi_{\text{OPT}(x)}$ (left) and $\Delta_\zeta = \zeta_{ICARO} - \zeta_{\text{OPT}(x)}$ (right) for $x = 60$ seconds and considering all the parameters range. We colored in blue the part of the curve where ICARO is achieving better results ($\Delta_\xi < 0$ or $\Delta_\zeta < 0$) and red otherwise. Again, ICARO clearly outperforms the optimal algorithm with limited prediction horizon when the mobility is pedestrian (average cost gap $E[\Delta_\xi] \approx -7.1\%$ average buffer under-run time gap

$E[\Delta_\zeta] \approx -0.02$), while the two solutions performs very close in the case of vehicular mobility ($E[\Delta_\xi] \approx 3.4\%$ and $E[\Delta_\zeta] \approx 0$).

Combining close to optimal performance and low complexity, ICARO shows that combining short term prediction with medium-long term statistical consideration makes for a robust solution in prediction-based resource optimization. Finally, compared to the wide-spread full buffer strategy an ICARO-based system is able to sustain the same quality of service while saving up 30% of the network resources or, analogously, 30% more users can be served with the same capacity.

Chapter 6

Anticipatory Quality-Resource Allocation for Multi-User Mobile Video Streaming

In this chapter, we study the network resource allocation problem aimed to minimize the average video re-buffering time (so called *lateness*) and, provided this objective is achieved, maximizing the average video bitrate for multiple users. We address video bitrate as a continuous quantity as in current streaming techniques, such as HTTP Live Streaming (HLS) [204], different segments can be encoded with different bitrates. Consequently, the average bitrate computed over multiple segments can take any value between the minimum and the maximum encoding bitrate.

Considering the video bitrate as a continuous quantity allows us to formulate the problem as a piecewise LP. In this LP, we i) minimize the aggregate lateness among all users when they are provided with the minimum allowed video bitrate and ii) the remaining resources are used to maximize the aggregate video encoding bitrate.

To solve this problem with a fast heuristic, we develop the Split, Sort & Swap (SS&S) algorithm. it achieves solutions very close to the optimum faster than standard solvers [205] for many relevant cases. In our evaluation of a 3GPP compliant macrocell scenario, SS&S never falls 0.5% below the optimum. This algorithm starts from a greedy solution obtained in polynomial time, which is subsequently refined by means of an iterative process. In each iteration, the algorithm provides a feasible solution that represents an improvement to the previous step. This property allows to trade-off algorithm runtime with quality gain. We believe that this practical trade-off and the ability to maximize quality while a minimum lateness is guaranteed, make our algorithm a promising candidate for a dedicated media streaming mode in fifth generation cellular networks. The algorithms low complexity supports online adaptation for a large number of users with a long prediction horizon.

In Section 6.1 we define the system model and discuss the optimization problem. In Section 6.2 we describe the SS&S algorithm, and we analyze its performance in Section 6.3.

6.1. Problem definition

In this paper we address resource allocation for the wireless downlink of a cellular network when future knowledge about the achievable data rate is available. To provide a simpler notation we will consider a system with a single base station to which all the K users connect. We call the set of users $\mathcal{U}$ and our prediction horizon is T time units and we refer to the set of time slots as $\mathcal{T}$. In the following, we consider unitary time unit $t = 1$, in order for data rates and download size to be used interchangeably. In the rest of the paper we use the following assumptions: 1) the future knowledge is perfect (this does not hold in practice, but the problem solution can be updated periodically. The present results can be considered as an upper bound for real scenarios); 2) the average video bitrate is continuous between 0 and q_M (e.g.: by combining segments of different quality) and 3) the quality of experience is proportional to the video bitrate.

The quantities of interest are:

- Per user achievable download rate $R = \{r_{i,j} \in [0, r_M],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where the entry $r_{i,j}$ represents the average rate that user i would achieve in slot j if he was using the cell alone. r_M is the maximum data rate of the specific mobile technology. This represents the future knowledge about network conditions, where slot 1 is the present slot.

- Minimum requirements $D = \{d_{i,j} \in [0, q_M],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where $d_{i,j}$ is the minimum amount of bytes user i should receive before the end of slot j in order to stream the video at the minimum allowed quality. If at any time the user receives more data than required, the excess can be stored in a buffer for later use.

- Assigned resources $A = \{a_{i,j} \in [0, 1],\ i \in \mathcal{U}, j \in \mathcal{T}\}$: each entry $a_{i,j}$ represents the average fraction of resources assigned to user i in slot j. In each slot, no user can be assigned more than the total available rate, $0 \leq a_{i,j} \leq 1$, nor can the sum of all the assignments exceed the total available resources in that slot, $0 \leq \sum_{i \in \mathcal{U}} a_{i,j} \leq 1$.

- Maximum extra video bitrate $U = \{u_{i,j} \in [0, q_M],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where $u_{i,j}$ is the additional bitrate user i could download before the end of slot j to increment the video quality.

- Assigned resources for extra quality $Q = \{q_{i,j} \in [0, 1],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where $0 \leq q_{i,j} \leq 1 - \sum_{k \in \mathcal{U}} a_{u,j}$ is the fraction of the available resources $r_{i,j}$ to be allocated for extra quality.

- Buffer state $B' = \{b'_{i,j} \in [0, b_M],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where

$$b'_{i,j+1} = [b'_{i,j} + a_{i,j} r_{i,j} - d_{i,j}]_0^{b_M} \tag{6.1}$$

is the buffer level of user i at the end of slot $j + 1$, b_M is the buffer size in bytes and $[\cdot]_a^b = \min\{\max\{\cdot, a\}, b\}$ is a bounding operator.

- Extra quality buffer state $B'' = \{b''_{i,j} \in [0, b''_{\max}],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where

$$b''_{i,j+1} = [b''_{i,j} + q_{i,j}r_{i,j} - u_{i,j}]_0^{b_M - b'_{i,j+1}} \tag{6.2}$$

is the buffered data to be used for extra quality. The total buffer is $B = B' + B''$.

- Lateness $L = \{l_{i,j} \in [0, 1],\ i \in \mathcal{U}, j \in \mathcal{T}\}$ is the fraction of slot j for which no data was available to stream the minimum video bitrate:

$$l_{i,j} = \begin{cases} [d_{i,j} - b_{i,j-1} - a_{i,j}r_{i,j}]_0/d_{i,j} & d_{i,j} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{6.3}$$

- Extra quality $E = \{e_{i,j} \in [0, q_M],\ i \in \mathcal{U}, j \in \mathcal{T}\}$, where $e_{i,j}$ is the maximum number of bytes that user i receives for extra quality j,

$$e_{i,j} = [q_{i,j}r_{i,j} + b_{i,j-1}]^{u_{i,j}}. \tag{6.4}$$

We define the system average lateness λ, and total average quality θ as:

$$\lambda = \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{T}} l_{i,j}/(KT). \tag{6.5}$$

$$\theta = \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{T}} (e_{i,j} + a_{i,j}r_{i,j})/T. \tag{6.6}$$

Thus we can define our optimization problem as computing two schedules A and Q the minimize lateness λ, and maximize quality θ, given C, D, U, and b_M as defined above. In this paper we assign a higher priority to lateness minimization so that under no circumstance the system is trading lateness for quality. Consequently we formulate the optimization problem as:

$$\begin{aligned} \underset{A,Q}{\text{minimize}} \quad & W\lambda - \theta \qquad (6.7) \\ \text{subject to:} \quad & a_{i,j} \geq 0;\ \sum_{k\in\mathcal{U}} a_{k,j} \leq 1 \\ & q_{i,j} \geq 0;\ \sum_{k\in\mathcal{U}} q_{k,j} \leq 1 - \sum_{k\in\mathcal{U}} a_{k,j} \\ & \forall i \in \mathcal{U}; j \in \mathcal{T} \\ & \text{Eqns. (6.1), (6.2), (6.3), (6.4), (6.5) and (6.6),} \end{aligned}$$

where Q and A are control variables, B', B'', L, E, λ and θ are additional variables and R, D, U and b_M are input parameters. The objective function is a linear combination of Eqns. (6.3) and (6.4) and the parameter W weights the two components. In particular, the solver has to use resources to either decrease the lateness or to increase the quality. Ideally, with $W \to \infty$ the solution of the problem would never choose quality over lateness, but in practice it is sufficient to have $W \gg \max\{\theta\}$.

In addition, since our objective function is a linear combination of piecewise linear equations, the overall optimization problem is piecewise linear as well. Unfortunately, while the formulation is quite compact, Eq. (6.1) that defines the evolution of the buffer state increases the overall complexity of the problem due to the bounding operator that limits the buffer between 0 and b_M in every slot.

6.2. Resource Allocation Algorithm

Our algorithm, Split, Sort & Swap (SS&S), is based on two main phases: first a greedy algorithm is used to obtain a feasible resource allocation (greedy phase), then the solution is iteratively improved (swap phase). The main idea is to i) compute the minimum lateness allocation and ii) maximize quality as second priority. According to our formulation this is equivalent to addressing the two quantities together since any lateness increment is W times worse than a similar decrease in quality. Also, this prevents mixing quality (video bitrate) and lateness (time). Thus, in the following, we provide a description of the algorithm operations to minimize lateness and we only discuss how to adapt it for quality maximization. The key ideas of the algorithm (from which we derived the name) are:

- **Split**: Consider the smallest number of slots.
- **Sort**: Use capacity in descending order.
- **Swap**: Change allocations only if it improves the objective.

The overall SS&S is given in Algorithm 3, using the notation from Section 6.1 and the following additional variables s_f and s_l that identify the first and the last slot the algorithm is considering at any given step. Variable s_l is increased whenever no more improvement to the objective function can be obtained to satisfy the requirements up to slot s_l. Variable s_f is increased if no improvement to the objective function can be obtained by changing the allocation earlier than s_f. Also, we define x and x_M as the current and the maximum allowed numbers of iterations of the greedy phase; λ_0 is the average lateness computed at the previous optimization iteration and δ_M is used to stop the greedy phase if the current improvement is smaller than that.

The following paragraphs review the algorithm's mechanics through a simple example, while its formal definition and the complete pseudocode is given shortly after. Let's consider the following achievable rates R (compare the topmost plot in Fig. 6.1), minimum video requirements

Algorithm 3 Split, Sort & Swap (SS&S)

```
Input: $R$, $D$, $b_M$.
Output: $[A, B] = \mathrm{SS\&S}(R, D, b_M)$
  $s_f = 1, s_l = 1$ // initial optimization window
  $a_{i,j} = 0, b_{i,j} = 0\ \forall i \in \mathcal{U}, j \in \mathcal{T}$
  while $s_l \leq T$ do
    $[A, B, s_f, s_l] = \mathrm{Greedy}(R, D, s_f, s_l, b_M, A, B)$
  end while
  $s_f = 1; x = 0; \lambda_0 = 0$
  while $x < x_M$ AND $|\lambda - \lambda_0| < \delta_M$ do
    $x = x + 1; \lambda_0 = \lambda$
    $[A, B, s_f] = \mathrm{Swap}(R, D, b_M, A, B, s_f)$
  end while
  return $A, B$
```

D and buffer size $b_M = 1$:

$$R = \begin{bmatrix} 2 & 0 & 3 & 0 \\ 1 & 1 & 4 & 1 \end{bmatrix} \tag{6.8}$$

$$D = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{6.9}$$

We start with the greedy procedure, the allocation A and lateness L of which are illustrated in the second and third plots of Fig. 6.1 respectively.

The first greedy allocation entails slot 1 only. Here the two users may obtain up to $r_{1,1} = 2$ and $r_{2,1} = 1$ respectively. The greedy procedure assigns $a_{1,1} = d_{1,1}/r_{1,1} = 1/2$ to satisfy slot 1 requirements. Then the remaining resources $a_{2,1} = 1 - a_{1,1} = 1/2$ are assigned to user 2. However, this is less than needed and causes a buffer under-run $l_{2,1} = d_{2,1} - a_{2,1}r_{2,1} = 1/2$. This under-run is unavoidable, due to the achievable rates in slot 1 and it obtains the minimum total lateness in slot 1 as increasing the resource allocation to user 2 would cause a larger lateness to user 1.

The greedy allocation in slot 2 is trivial for two reasons: slot 1 is fully allocated, thus no buffering is possible, and $r_{1,2} = 0$. Thus, we obtain $a_{1,2} = 0$, $a_{2,2} = 1$, $l_{1,2} = 1$ and $l_{2,2} = 0$ respectively. We annotated these two first slots as G1 in the figure and the total lateness so far is $\lambda = l_{2,1} + l_{1,2} = 3/2$.

The allocation is now complete until slot 2, since no other greedy allocations can occur before and including slot 2, thus the next optimization phase will have $s_f = s_l = 3$. We annotated the following two slots ($j = \{3, 4\}$) as G2 in Fig. 6.1. Again, the greedy operation starts by considering slot 3 alone by sorting users according to their achievable rates. Thus, slot 3 requirements are satisfied by allocating $a_{1,3} = 1/4$ and $a_{2,3} = 1/3$ and leaving $5/12$ of the resources free. Now the algorithm accounts for slot 4 requirements by considering free resources both in slot 3 and

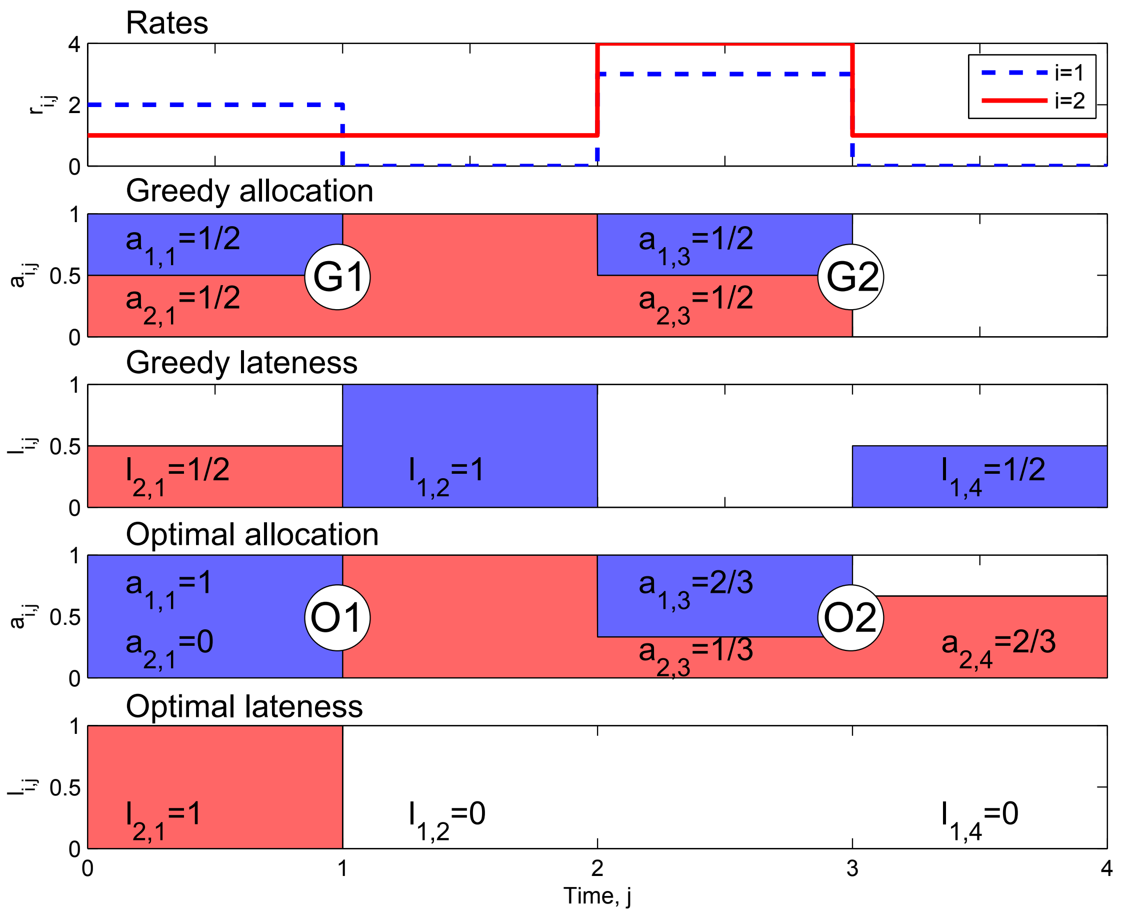


Figure 6.1: Graphic example of simple SS&S operations: users' achievable rates, R are shown at the top, the allocation A and the lateness L after the greedy phase are plotted in the second and third plots from the top respectively. While the fourth and the fifth plot show the optimal A and L after the swap phase. The improvements from G1 to O1 and from G2 to O2 involve a Type 1 and a Type 2 swap respectively.

slot 4 in order of decreasing capacity. The following greedy allocations are made: since $r_{1,3} = 4$ is the highest, $a_{1,3} = a_{1,3} + d_{1,4}/r_{1,3} = 1/4 + 1/4 = 1/2$ of which $1/4$ is used to fill the buffer $b_{1,3} = b_{1,2} + a_{1,3}r_{1,3} - d_{1,3} = 1$; the highest capacity for user 2 is in slot 3 too, but here user 2 can only be assigned $a_{2,3} = 1/3 + 1/6 = 1/2$ to buffer $b_{2,3} = 1/2$ and, since user 2 capacity in the last slot is $r_{2,4} = 0$, the greedy decision cannot avoid some lateness $l_{2,4} = d_{2,4} - b_{2,3} = 1/2$.

The final greedy allocation is obtained with a total lateness $\lambda = 2$ consisting of an unavoidable under-run in slot 1 for user 2, two under-runs for user 1 in slots 2 and 4 and all resources in the last slot left unassigned.

In the swap phase, the algorithm considers those slots where it was not possible to avoid some lateness by modifying the greedy allocation obtained so far. In our example, the first case to be addressed is user 1 in slot 2, where the algorithm obtained a lateness of $l_{1,2} = 1$. If we do not consider user 2, user 1 can fill the buffer in slot 1 to satisfy the requirements of slot 2.

However, this cause more lateness for user 2 in slot 1. In particular, if we swap a quantity δ_a of resources from user 2 to user 1, we obtain that the total lateness varies of a proportional quantity $\delta_\lambda = \delta_a(r_{2,1} - r_{1,1})$ that is the sum of the increase of lateness $l_{2,1} = l_{2,1} + \delta_a r_{2,1}$ and the decrease of $l_{1,1} = l_{1,1} - \delta_a r_{1,1}$ due to the resource swap.

If $r_{1,1} > r_{2,1}$, then $\delta_\lambda < 0$ and the total lateness is decreasing. However, δ_a is limited by the minimum among $a_{2,1} = 1/2$ the resources allocated to the users we are removing resources from, $(b_M - b_{1,1})/r_{1,1} = 1/2$ the maximum resources that can be buffered by the receiving user and $l_{1,2}/r_{1,1} = 1/2$ the lateness we are trying to reduce. Since $\delta_\lambda = -\delta_a$, the maximum improvement is obtained for the maximum $\delta_a = 1/2$ and the following optimal allocation $a_{1,1} = 1$, $a_{1,2} = 0$, that allows user 1 to buffer $b_{1,1} = 1$ and avoid the under-run in slot 2 and increase the under-run for user 2 to $l_{2,1} = 1$. However, the total lateness of these first two slots is $\lambda = 1$ only: δ_λ less than that obtained by the greedy procedure. We annotate these two slots $j = \{1, 2\}$ as O1 in the figure.

The forth and the fifth plot in Fig. 6.1 show the optimal allocation and associated lateness obtained after the swap phase. We call **Type 1** the swap occurring in the first two slots from G1 to O1 and is characterized by replacing an under-run with a smaller one earlier in the sequence.

Before addressing the buffer under-run in slot 4, we note that in the two first slots all resources are given to the user with the highest achievable rate, thus further resource swapping can only make the total lateness worse. Also, applying the type 1 method to this is not worthy since $r_{1,3} < r_{2,3}$ and will lead to an increased lateness $l_{2,3} > l_{1,4}$. However, slot 4 has unused resources and it may be possible to use them to improve the earlier allocations.

In fact, since slot 4 requirements for user 2 are satisfied with buffered data, it is possible avoid buffering for user 2 and, instead, use the free resources in the last slot to satisfy $d_{2,4}$, while the remaining resources in slot 3 can be buffered for user 1 to satisfy $d_{1,4}$. More formally, we can swap a quantity of resource δ_a from user 2 to 1, which will cause a proportional lateness variation $\delta_\lambda = \delta_a(r_{2,3} - r_{1,3})$ and we can recover $\delta_a r_{2,3}$ by allocating $a_{2,4} = \delta_a r_{2,3}/r_{2,4}$.

Similar to a Type 1 swap, δ_a is limited by the minimum among $a_{2,3} = 1/2$, $(b_M - b_{1,3})/r_{1,3} = 1/3$ and $l_{1,4}/r_{1,3} = 1/3$ and, in addition, by the free resources in slot 4 $(1 - \sum_{i \in \mathcal{U}} a_{i,4})/r_{1,3} = 1/3$. Thus the optimal allocation becomes $a_{1,3} = 2/3$, $a_{2,3} = 1/3$, and $a_{2,4} = 2/3$. These last two slots are annotated as G2 in the figure and we call **Type 2** the resource swap between G2 and O2, which is defined by the recovery of an under-run by modifying earlier buffer state with the usage of later free resources.

More formally the greedy and the swap phases are given in Algorithm 4 and Algorithm 6 respectively. In Algorithm 4 we use the indicator function $\mathrm{I}(x) = 1$ if $x > 0$ and $\mathrm{I}(x) = 0$ otherwise. Also, while the greedy phase is deterministic and performs at most $O((KT)^2)$ iterations, the swap phase improves the objective functions iteratively.

For what concerns the greedy phase, Algorithm 4 checks whether further resources can be assigned between slot s_f and s_l by computing $\hat{A}$, which elements $\hat{a}_{i,j}$ represent the maximum usable rate for user i in slot j accounting for available resources in the slot $(1 - \sum_{k \in \mathcal{U}} a_{k,j})$, future requirements to be satisfied ($\sum_{k=j}^{s_l} d_{i,k}$) and available buffer space ($b_M - \max_{k\in[j,s_l]}\{b_{i,k}\} + d_{i,j}$) as follows:

$$\hat{a}_{i,j} = \min\Big\{(1 - \sum_{k\in\mathcal{U}} a_{k,j})r_{i,j},\ \sum_{k=j}^{s_l} d_{i,k},\ b_M - \max_{k\in[j,s_l]}\{b_{i,k}\} + d_{i,j}\Big\} \quad \forall\, i \in \mathcal{U}, s_f \le j \le s_l. \tag{6.10}$$

Algorithm 4 Greedy phase

```
Input: $R, D, s_f, s_l, b_M, A, B$.
Output: $[A, B, s_f, s_l] = \mathrm{Greedy}(R, D, s_f, s_l, b_M, A, B)$
  compute $\hat{A}$ as per Eq. (6.10) // feasible resource usage
  while $\sum_{i\in\mathcal{U}}\sum_{j=s_f}^{s_l} > 0$ do
    $k, l = \operatorname{argmax}_{i,j}\, r_{i,j}\mathrm{I}(\hat{a}_{i,j})$ // choose best user and slot
    $a_{k,l} = a_{k,l} + \hat{a}_{k,l}/r_{k,l}$ // increase allocation
    $b_{k,m} = b_{k,m} + \hat{a}_{k,l}\ \forall\, l \le m < s_l$ // adjust buffer
  end while
  $s_l = s_l + 1$
  $s_f = \mathrm{NewStart}(A, B, R, b_M, s_f, s_l)$
  return $A, B, s_f, s_l$
```

Then, among all users and slots to whom resources can be assigned ($\hat{a}_{i,j} > 0$) the user k with the highest rate in slot l is assigned further resources, so that $a_{k,l} = a_{k,l} + \hat{a}_{k,l}/r_{k,l}$. One greedy phase step continues until either no new resources can be assigned or all requirements up to slot s_l are satisfied.

Finally, s_l is increased and the $\mathrm{NewStart}$ procedure (see Algorithm 5) updates s_f if needed. In particular, a slot s_f is completed if it is not possible to swap resources between users either because that would cause a buffer overflow or because that would degrade the objective function.

Algorithm 5 New Start

```
Input: $A, B, R, b_M, s_f, s_l$.
Output: $s_f = \mathrm{NewStart}(A, B, R, b_M, s_f, s_l)$
  for $j \in [s_f, s_l]$ do
    if $\min\{b_M - \max_{l\in[j,s_l]} b_{i,l},\ \sum_{k\in\mathcal{U}|r_{k,j}<r_{i,j}} a_{k,j}\} = 0 \quad \forall\, j \in \mathcal{U}$ then
      $s_f = s_f + 1$
    else
      return $s_f$
    end if
  end for
  return $s_f$
```

The formal definition of the swap phase requires first to generalize the two type of resource swapping that, in turn, requires to identify the best possible swap which is improving the objective function the most by increasing the resource allocation the least.

According to type 1 swap, the best swap is the one obtaining the highest δ_λ. To this extent, we first define $\delta_{\lambda,(i,j)}$, and $\delta_{a,(i,j)}$ as the best lateness improvement and the maximum exchangeable resources to move an under-run in (i,j) to $n_{i,j}$, where

$$n_{i,j} = (m,n) = \operatorname*{argmax}_{l \in \mathcal{U}, k<j} \delta_{a,(i,j)}(r_{i,k} - r_{l,k}) + \delta_{\lambda,(l,k)} \tag{6.11}$$

$$\delta_{\lambda,(i,j)} = \delta_{a,(i,j)}(r_{i,n} - r_{m,n})\delta_{\lambda,(m,n)} \tag{6.12}$$

$$\delta_{a,(i,j)} = \min\{a_{m,n},\ (b_M - \overline{b}_{i,n\to j})/r_{i,n},\ \delta_{\lambda,(i,j)}/r_{i,n}\}, \tag{6.13}$$

and $\overline{b}_{i,n\to j} = \max_{k\in[n,j]}\{b_{i,k}\}$ is the maximum buffer state for user i from slot n to j. Computing Eqns. (6.11), (6.12) and (6.13) is recursive as chained swapping is also considered and can be computed from $j = s_f$ to s_l after initializing $n_{i,j} = (i,j)$, $\delta_{\lambda,(i,j)} = 0$ and $\delta_{a,(i,j)} = 1$.

The best type 1 swap (i^*, j^*) among all the possible (l,k) (there is an under-run ($l_{l,k} > 0$ to be solved and resources can be swapped $\delta_{a,(l,k)} > 0$) is

$$(i^*, j^*) = \operatorname*{argmax}_{(l,k)\,\text{s.t.}\,(l_{l,k}>0 \wedge \delta_{a,(l,k)}>0)} \delta_{\lambda,(l,k)} \tag{6.14}$$

and can be resolved by following the chain starting from (i^*, j^*) which moves the under-run to $(m,n) = n_{i^*,j^*}$ by swapping $\delta_{a,(i^*,j^*)}$ resources from user m to user i^* in slot n. Then, $(i^*, j^*) = (m,n)$ and the next chained swap is resolved until $n_{i^*,j^*} = (i^*, j^*)$, which means no more swaps can be done to reduce an under-run in the last (i^*, j^*).

Conversely, type 2 swaps are defined as those that reduce the total lateness by by modifying earlier buffer states exploiting later free resources. The best type 2 swap is the one obtaining the highest product between exchangeable resources and lateness decrease $\delta_a\delta_\lambda$. To this extent, we redefine $\delta_{\lambda,(i,j)}$, $\delta_{a,(i,j)}$ and $n_{i,j}$, but in this case $(m,n) = n_{i,j}$ is where new resources are allocated to compensate for the swap from user m to i in slot j:

$$n_{i,j} = (m,n) = \operatorname*{argmax}_{l\in\mathcal{U}, k>j} \delta_{a,(l,k)}\delta_{\lambda,(l,k)} \tag{6.15}$$

$$\delta_{\lambda,(i,j)} = \delta_{\lambda,(m,n)} \tag{6.16}$$

$$\delta_{a,(i,j)} = \min\{\delta_{a,(m,n)} r_{m,n}/r_{m,j},\ a_{m,j},\ \underline{b}_{m,n\to j})/r_{l,n}\} \tag{6.17}$$

where $\underline{b}_{m,n\to j} = \min_{k\in[n,j-1]}\{b_{m,k}\}$ is the minimum buffer level for user m from slot n to $j-1$ and ensures that it is possible to reduce the buffer allocation to save resources in slot j. Computing Eqns. (6.15), (6.16) and (6.17) is recursive as chained swapping is also considered and can be computed by going backwards from $j = s_l$ to s_f after initializing $n_{i,j} = (i,j)$,

$\delta_{\lambda,(i,j)} = I(1 - \sum_{k\in\mathcal{U}} a_{k,j})r_{i,j}$ and $\delta_{a,(i,j)} = 1$. The best type 2 swap (i^*, j^*) among all the possible (l, k) (under-run in $l_{l,k} > 0$ and $\delta_{a,(l,k)} > 0$) is

$$(i^*, j^*) = \underset{(l,k)\,\text{s.t.}\,(l_{l,k}>0\wedge\delta_{a,(l,k)}>0)}{\operatorname{argmax}} \delta_{a,(l,k)}\delta_{\lambda,(l,k)} \tag{6.18}$$

and can be resolved by allocating new resources in $\delta_{a,(i^*,j^*)}$ and following the chain moving free resources to $(m, n) = n_{i^*,j^*}$ by swapping $\delta_{a,(i^*,j^*)}$ resources from user i to user m in slot n. Then, $(i^*, j^*) = (m, n)$ and the next chained swap is resolved until $n_{i^*,j^*} = (i^*, j^*)$, where the under-run in the last (i^*, j^*) is decreased.

Finally, Algorithm 6 lists the steps of the swap phase. Basically, each iteration checks whether a Type 1 or a Type 2 swap can be done and, in the positive case, it recomputes the allocation. Type 2 swaps are prioritized over Type 1, since it is more efficient to use remaining resources first and reducing lateness later. In order to use the SS&S algorithm to solve the quality maximization problem, we can simply observe that the resources allocated for extra quality Q cannot modify those assigned to minimum requirements A, and the buffered data for extra quality must account for data already buffered for minimum requirements $B'' < b_M - B'$. Thus running $[Q, B''] = \text{SS\&S}(R, U, b_M - B')$ will provide the desired solution.

6.3. Simulation Results

This section evaluates the performance of SS&S and its result after x iterations (SS&S(x)) against the optimal solution (Optimum) and the unoptimized performance (Baseline). The baseline is computed assuming proportionally fair scheduling is allocating resources [197] by allowing each users $1/K$-th of the time. While, the optimal performance is obtained as the solution of the optimization problem in Eq. (6.7) by means of standard solvers, such as GUROBI [205].

Each simulation is run over traces generated according to the LTE model in [197] assuming random cell deployment with average cell distance of 500 meters, users moving according to a random waypoint mobility model with an average speed of 10 meters per second, 10 active users are considered in each simulation and the video is 180 seconds long. In each trace the capacity oscillates according the the distance from the users to the base station and has 4 maxima, which are approximately 50 seconds apart from each other.

In each simulation, all traces are normalized so that the average capacity is $C = 1$ and, thus, equal for all users. In all simulation the buffer size is set to last at least 50 seconds at the maximum quality, so that a full buffer allow a user to playing the video without interruptions even if no download is made between two capacity maxima. Finally, each parameter combination is averaged over 50 runs and error bars are plotted for this averages at 95% confidence.

In order to systematically study various rate requirements for video streaming, we define two additional parameters: $\alpha \in (0, \infty)$ and $\beta \in [0, 1]$. The parameter $\alpha = K(d_{i,j} + u_{i,j})/C$ represents the maximum video bitrate requested by all users for every user i and slot j. Even

Algorithm 6 Swap phase

```
Input: $R, D, s_f, b_M, A, B$.
Output: $[A, B, s_f] = \text{Swap}(R, D, s_f, b_M, A, B)$
  if Type 2 then
      $(l, k) = (i^*, j^*)$ as per Eq. (6.18)
      $\delta_a = \min\{\delta_{a,(l,k)},\ l_{l,k}/r_{l,k}\}$
      while $n_{l,k} \neq (l, k)$ do
          $(m, n) = n_{k,l}$
          $a_{l,k} = a_{l,k} + \delta_a$
          $a_{m,k} = a_{m,k} - \delta_a$
          $\delta_a = \delta_a\ r_{m,k}/r_{m,n}$
          $(l, k) = (m, n)$
      end while
      $a_{l,k} = a_{l,k} + \delta_a$
      Recompute $B$ as per Eqns. (6.1)
  else if Type 1 then
      $(l, k) = (i^*, j^*)$ as per Eq. (6.14)
      $\delta_a = \min\{\delta_{a,(l,k)},\ l_{l,k}/r_{l,n}\}$
      while $n_{l,k} \neq (l, k)$ do
          $a_{l,n} = a_{l,n} + \delta_a$
          $a_{m,n} = a_{m,n} - \delta_a$
          $(l, k) = (m, n)$ and $(m, n) = n_{l,k}$
          $\delta_a = \delta_a r_{m,n}/r_{m,k}$
      end while
      recompute $B$ as per Eqns. (6.1)
  end if
  $s_f = \text{NewStart}(A, B, R, b_M, s_f, T)$
  return $A, B, s_f$
```

though minimum and maximum requirements are assumed constant for the whole video and equal across the users, this only simplifies the numerical study and does not limit for the algorithm. Also, $\alpha = 1$ means that the demand is equal to the average offer. Instead, $\beta = d_{i,j}/(d_{i,j} + u_{i,j})$ represents the ratio between the minimum and the maximum video bitrate. Thus, $0 < \beta < 1$ means that the video quality may be as low as β in order to stream the video without interruptions.

Fig. 6.2 shows the first set of plots, that illustrate, from the left to the right, the average lateness λ, the average total video quality θ normalized over the average capacity and the gap between the results obtained by SS&S and the optimal.

The first plot is obtained with $\alpha \in [0.25, 10]$ in logarithmic steps and $\beta = 1$. This setup is meant to study λ as a function of the ratio between demand and offer, thus no extra quality is considered. The results obtained by SS&S and the optimal are plotted as black solid and blue dash-dotted lines respectively and they are very close to each other confirming that SS&S obtains almost optimal performance. SS&S(x) performance are plotted for $x \in [1, 1000]$ as dotted black lines and they are increasingly close to the optimal performance with increasing x. Finally, the

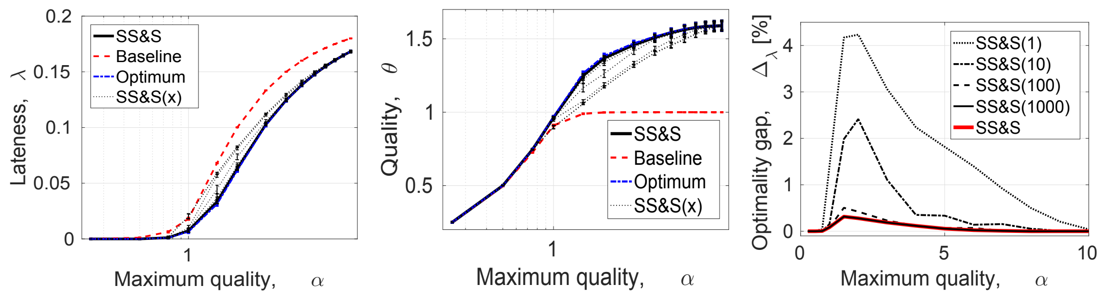


Figure 6.2: Performance comparison among SS&S, optimal and baseline: the plots show the average lateness, the average total video quality normalized over the average capacity and the gap between the results obtained by SS&S and the optimal varying the number of iterations.

baseline performance is shown as a red dashed line. Notably, the baseline performance is always worse than the others and it is obtaining an average lateness more than twice (2.45) as long as SS&S when $\alpha = 1$.

The second plot, in the center, is obtained for $\beta = 0$ with everything else unchanged. This set of experiments is meant to study the maximum average bitrate achievable varying α. Again SS&S reaches almost optimal performance and both solutions outperform the baseline by up to 60% and, starting from $\alpha \leq 1.5$ of as much as 25%. As in the previous graph, increasing the number of iterations reduces the distance from SS&S(x) to the optimum. Notably, both here and in the previous graph, a larger number of iterations is needed for $\alpha \in [1, 3]$: in fact, out of this region a completely greedy solution is already good enough as the minimum requirements are either very low ($\alpha < 1$) and they can be greedily allocated to all the users or very high ($\alpha > 3$) so that only the user with the highest capacity is being allocated.

The third plot of Fig. 6.2 shows the difference between the lateness obtained by SS&S(x) and the optimal $\Delta_\lambda = \lambda_{\text{SS\&S}(x)} - \lambda_{\text{Opt}}$ varying $x \in \{1, 10, 100, 1000\}$. Again the gap is larger for fewer iterations and in the region $1 < \alpha < 3$. Also, for $x >= 1000$ are sufficient to achieve the best performance of SS&S, which, in turn, are very close to the optimal ($\Delta_\lambda < 5 \cdot 10^{-3}$). Finally, even with a single iteration the gap is smaller than 5% ($\Delta_\lambda \approx 0.043$).

The second series of plot in Fig. 6.3 represents from left to right: contour plots of the average lateness, contour plots of the total average quality and the trade off between lateness and quality varying both $\alpha \in [1, 10]$ and $\beta \in [0.1, 0.9]$. In the first two plots a black dashed contour is plotted to mark the boundary between the region where minimum requirements for an uninterrupted streaming are lower (bottom-left part) or larger (upper-right part) than the average capacity.

The lateness results (left) are quite intuitive as below the dashed border SS&S mostly streams the video uninterrupted at the desired minimum quality. However, crossing this border causes an increasingly higher lateness up to 20% of the video duration. Conversely, the quality contours (center) are slightly more complex: in fact the quality increases both above and below the dashed border. The quality increase when the resources are scarcer (top-right part) is justified by the fact that the system is trading lateness for quality allocating only users that can obtain higher quality.

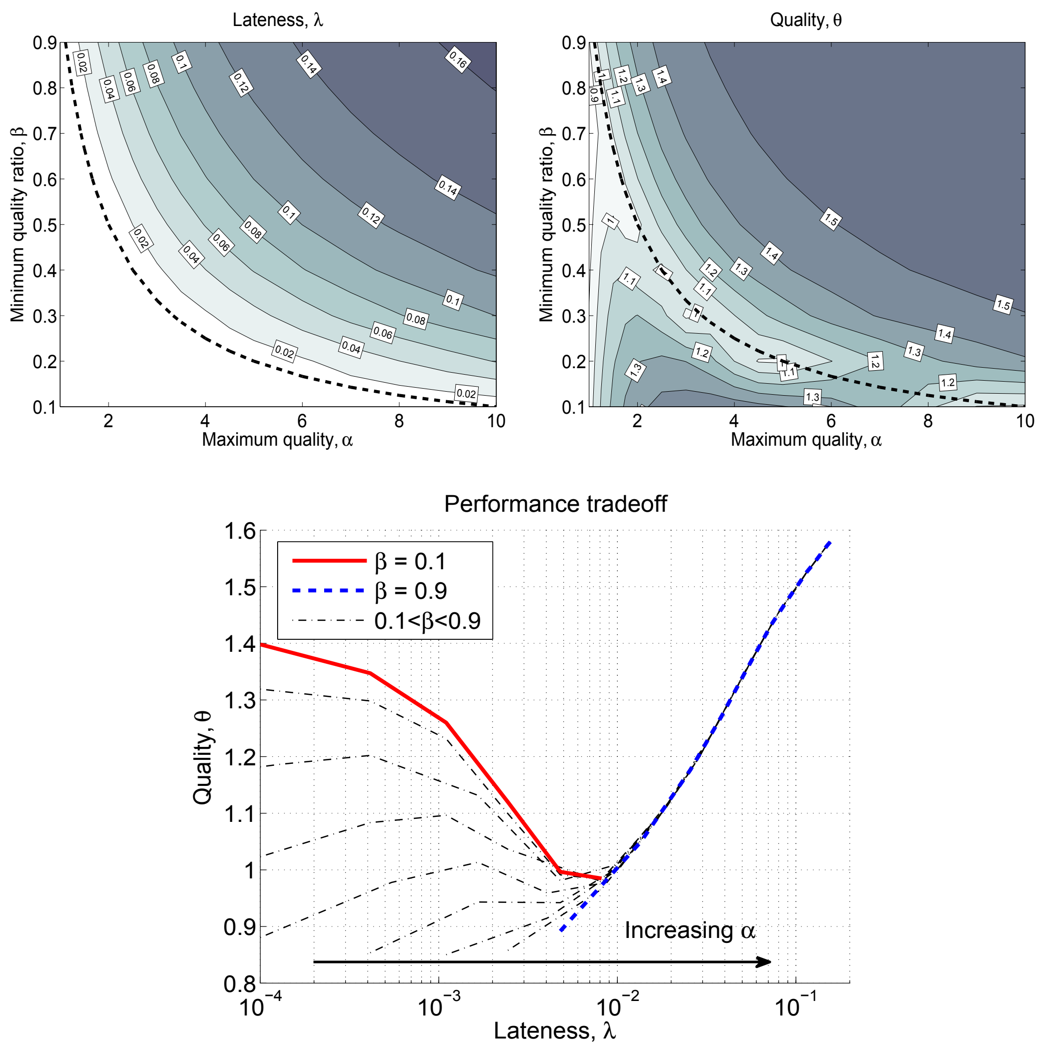


Figure 6.3: Contour plots of the average lateness (left), average total quality (center) and trade off plot (right) between lateness and quality varying α and β.

Conversely, the quality increase in the lower-left part of the figure is obtained without almost no lateness (compare to the left figure): in fact, in this region the minimum requirements $d_{i,j}$ are small compared to the average capacity, thus allocation computed by SS&S allow all the users to receive an uninterrupted stream at a quality which is at least equal to $\alpha\beta C/K$.

The trade off between lateness and quality is clearly illustrated in the right plot if Fig. 6.3, where λ and θ are plotted on the x and y axes respectively. The different curves are plotted for different β and each curve is obtained for varying α. Notably, the system is bound between $\beta = 0.1$ on the top-left part and $\beta = 0.9$ on the right. All the curves are quite close when $\alpha\beta = 1$. This plot can be used to estimate the expected system performance when adopting SS&S: for instance, in a system where the minimum requirements are 20% of the maximum quality (e.g.: 400 kbps and 2 Mbps) the second curve from the top can be used to decide how many users to allow in the system as a function of the desired average video bitrate; as an example, if $C = 20$ Mbps and the desired average video bitrate should not be lower than 1.5 Mbps, the user number should be $K \approx 16$, obtaining an average lateness lower than $\lambda < 10^{-4}$; more users can be traded for lower average quality, lower minimum quality or higher lateness.

Chapter 7

Anticipatory Admission Control and Resource Allocation for Media Streaming in Mobile Networks

In this chapter we investigate prediction based media streaming in mobile networks and we discuss admission control and resource allocation. The quality of a media stream is characterized by the following Key Performance Indicators (KPIs) [206]: (i) streaming continuity and (ii) average stream quality. The former is assumed to have higher priority, since in general interruptions may jeopardize the comprehension of the content and therefore are perceived as the worst quality degradation. The latter is optimized with lower priority, since, even if it has a weaker impact on user's perception, users appreciate when a certain agreed QoS is guaranteed. In this paper we consider it to be directly proportional to the stream bitrate [207].

An additional characteristic of prediction based optimization is that the prediction reliability varies in time and, usually, decreases as the prediction horizon length grows (see Chapter 3). Therefore, anticipatory optimization schemes should consider this either explicitly in the problem formulation [39] or evaluate the impact of prediction error a posteriori [52]. Here we focus on joint admission control and resource allocation with perfect system state prediction to obtain upper bounds on the achievable gains.

We follow a lexicographic approach where, first, we maximize the number of users that are served with guaranteed QoS for the whole duration of the media stream, minimizing the total interruption time, and maximizing the streaming quality. Thus, the streaming requests that cannot be scheduled with guaranteed quality must wait for the system to have enough resources for them to start streaming. Furthermore, we assume that it is always preferable to admit a new user in the system than increasing the quality of a user who is already admitted and the streaming continuity is always preferred to extra quality.

We validate our approach using trace based simulation obtained from real measurement data collected by the MOMENTUM project [160] in Berlin. We show that our online solution closely approximates the results achieved by the MILP formulation and dramatically reduces the computational time.

The rest of the chapter is structured as follows: section 7.1 reviews the state of the art on anticipatory networking solutions, section 7.2 introduces the mathematical notation and the optimization problem, section 7.3 describes our proposed approximate solution, and section 7.4 illustrates our evaluation campaign.

7.1. Related work

Anticipatory optimization techniques are motivated by a series of seminal papers, such as [109, 112], which discuss the predictability of human mobility patterns and the link between mobility and communication. Shafiq et al. [112] studied mobile network traffic and its spatio-temporal correlation with mobility patterns. Similarly, Ahmed et al. [202] studied network user habits in terms of content: the study links content requests and user categories, aiming to their prediction.

The predictability of network capacity and the achievable rate of mobile users have been extensively studied in the literature. These studies range from short term prediction using filtering techniques [103, 192], to medium and long term forecasting solutions [35, 81] accounting for position and trajectory estimates. We contributed to the literature with a general model [4] for predicted rates in mobile networks accounting for prediction uncertainties, and we use the model to devise single user optimal resource allocation policies [6].

For what concerns the state of the art on prediction based network optimization, in what follows we review a few of the papers that are more closely related to our current work.

Majid et al. [208] and Koutsakis et al. [209] exploited medium-long term average prediction of the users' achievable rate to devise call admission control and resource allocation techniques, respectively. While the former is more focused on Differentiated Services (DiffServ) system [210], the latter addressed specifically multimedia traffic in broadband mobile networks. The present work differs from these early papers as well as more recent approaches [211], since we exploit rate fluctuations on a shorter time scale instead of using averages.

More recently, Dräxler and Karl [54] tackled multimedia traffic optimization by devising a different problem formulation that considered an objective function that combined stream interruption time and average quality. The proposed schemes choose when to download a given content segment and at which quality among a discrete set of qualities. In this paper we obtain a simpler formulation by considering continuous quality and by means of approximations. This allows us to include in our objective function both admission control and resource allocation.

Abou-zeid et al. [22, 52] develop a MILP formulation of a similar problem to obtain an optimal resource allocation and to increase energy efficiency. As other prior work, these papers do not consider admission control and thus they cannot enforce QoS in the system. A different approach is taken in [212] and [7], which study different algorithms to solve the resource allocation problem. These approaches aim at finding practical solutions that do not require commercial solvers and can execute in real-time even with non-linear objective functions. In addition, complete solutions, such as [39], integrate prediction techniques and optimization algorithms to solve the resource allocation problem or study optimal video transcoding [213] for admission control and scheduling.

Compared to the aforementioned solutions, this paper proposes a different perspective of the network optimization problem as we enforce QoS by means of admission control. In addition, we propose low-complexity solutions that can be used for online optimization, which require the output to be updated within a short time.

7.2. Problem Definition

The admission control and resource allocation problem can be modeled as a centralized decision making problem, where a set $\mathcal{N}$ of N users share a given quantity of network resources. Prediction is assumed to be perfect over a set $\mathcal{T}$ of T time slots. In the following, we consider slot duration $t = 1$, thus data rate and download size can be used interchangeably. In the rest of the paper we use the following assumptions: (a) the future knowledge is perfect and (b) the average video bitrate is continuous between 0 and q_M (e.g., by averaging over segments of different quality [204]).

- Predicted achievable download rate $r_{i,j} \in [0, r_M]$ is the prediction of the rate a user would achieve if no other user is scheduled. r_M is the maximum achievable data rate.

- Minimum requirement $d_{i,j} \in [0, q_M]$ is the minimum amount of bytes needed in a given slot to stream the content at the minimum bitrate with no interruptions.

- Maximum extra video bitrate $u_{i,j} \in [0, q_M]$, is the maximum amount of additional bytes that can be used in a given slot to obtain the maximum content bitrate.

The problem is characterized by the following variables:

- Resource assignment $a_{i,j} \in [0, 1]$ represents the average fraction of resources assigned to user i in slot j. In each slot, each user can be assigned at most the total available rate, $0 \leq a_{i,j} \leq 1$, and the sum cannot exceed the total available resources, $0 \leq \sum_{i \in \mathcal{N}} a_{i,j} \leq 1$. Figure 7.1 shows an example with $N = 3$ and $T = 20$. In the top graph the achievable rates are plotted independently. In the center plot, a possible resource assignment is visualized by stacking the fraction of resources assigned to each of the users $a_{i,j}$ on top of each other. In the bottom graph, the cell capacity variation is addressed by stacking the product of the achievable rate and the fraction of assigned resources $a_{i,j} r_{i,j}$.

- Buffer state $b_{i,j} \in [0, b_M]$ tracks the amount of bytes stored in the buffer and b_M is the buffer size in bytes.

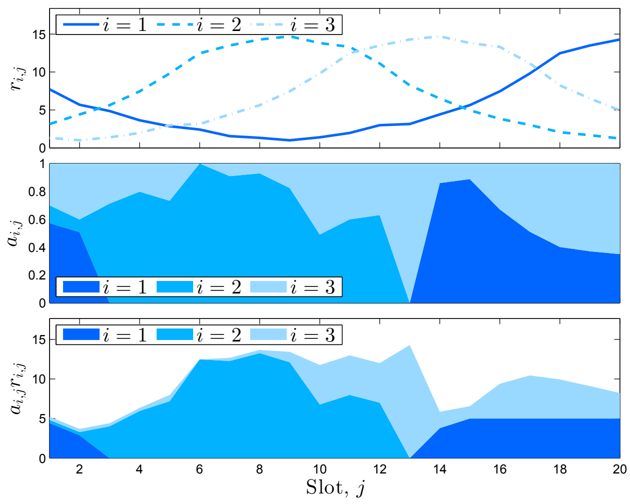


Figure 7.1: An example of achievable rates $r_{i,j}$ (top), assignments $a_{i,j}$ (center) and obtained rates $a_{i,j}r_{i,j}$ (bottom) in a 3-user scenario.

- Pre-buffering time (or waiting time) $w_{i,k} \in \{0,1\}$ with $k \in \{1,\dots,T+1\}$ defines when the actual playing of the content starts: there must be a single starting point ($\sum_{k=1}^{T+1} w_{i,k} = 1, \forall i \in \mathcal{N}$). Thus user i will wait for $W_i = (\text{argmax}_k w_{i,k}) - 1$ slots where she can only fill the buffer. This waiting implies the requirement sequence has to be shifted to later slots. Thus, in slot j user i is obtaining the rate $a_{i,j}r_{i,j}$ and should satisfy the shifted requirements $\overrightarrow{d_{i,j}} = \sum_{k=1}^{T+1} D_{i,j,k}w_{i,k}$ and $\overrightarrow{u_{i,j}} = \sum_{k=1}^{T+1} U_{i,j,k}w_{i,k}$, where D and U are $N \times T \times T+1$ matrices whose vectors $\mathbf{d_{i,k}} = \{\mathbf{0_{k-1}}, d_{i,1}, \dots, d_{i,T-k}\}$ and $\mathbf{u_{i,k}} = \{\mathbf{0_{k-1}}, u_{i,1}, \dots, d_{i,T-k}\}$ are shifted versions of the original requirements, where we used bold fonts to identify vectors and $\mathbf{0_k}$ is a null vector of size k.

- Interruption time[1] (or lateness) $l_{i,j} \in [0, q_M]$ is the missing data to fulfill the minimum content requirement $\overrightarrow{d_{i,j}}$:

$$l_{i,j} = [\overrightarrow{d_{i,j}} - b_{i,j} - a_{i,j}r_{i,j}]_0^{\overrightarrow{d_{i,j}}} \tag{7.1}$$

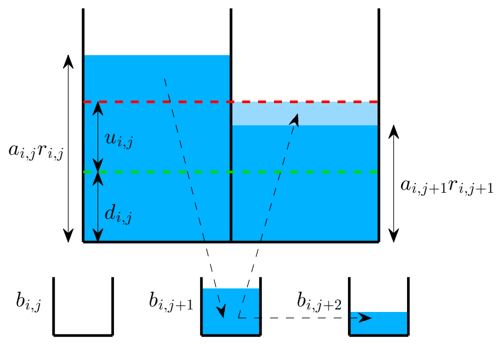
where $[x]_a^b = \min\{\max\{x,a\},b\}$ is a bounding operator that forces the undelivered quantity to be greater than zero and smaller than the requirement in the slot.

- Extra quality outage $e_{i,j} \in [0, q_M]$ is the amount of data missing to obtain the content at the maximum bitrate $\overrightarrow{u_{i,j}}$,

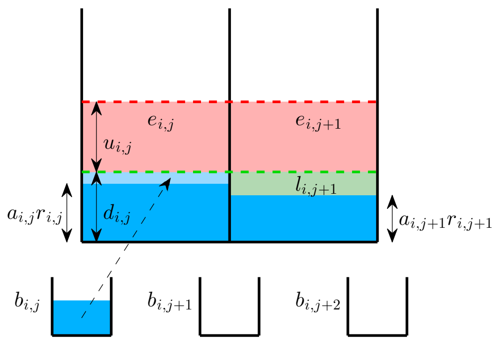
$$e_{i,j} = [\overrightarrow{u_{i,j}} + \overrightarrow{d_{i,j}} - l_{i,j} - b_{i,j} - a_{i,j}r_{i,j}]_0^{\overrightarrow{u_{i,j}}}. \tag{7.2}$$

Figure 7.2(a) provides a graphical example of the buffer usage for a single user over two subsequent slots. Starting from an empty buffer, the obtained rate $a_{i,j}r_{i,j}$ is used to satisfy the

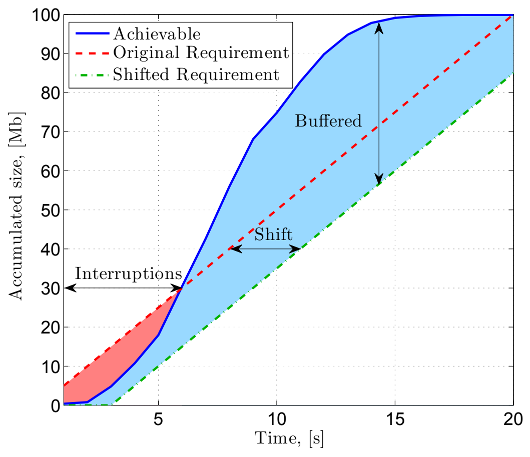
[1] Since receiving less data than the minimum requirement causes an interruption in the streaming, we use the effect instead of the cause to define this quantity. However, the actual interruption time is the ratio between missing and minimum requirement in a slot.



(a) Buffer



(b) Outage



(c) Shift

Figure 7.2: Three examples of the system quantities: 7.2(a) exemplifies the buffer usage over two subsequent slots; 7.2(b) shows lateness and extra quality outage examples; 7.2(c) illustrates the impact of pre-buffering.

current requirements and to buffer content for the next slot. The light area of the second slot highlights the fraction of content that has been previously buffered. Whether the buffer contains data to guarantee continuous streaming or extra quality is a key decision in the system and plays a critical role in the following optimization.

Figure 7.2(b) shows a two slot example where the user does not obtain a rate sufficient to satisfy the requirements: in the first slot this is compensated by the buffer, but this is not possible in the second slot resulting in an interruption of the streaming. Thus, the figure shows in light red the quality outage and in light green the missing minimum requirements in the second slot.

Figure 7.2(c) shows the cumulative download size and requirements according to the second user of the example of Figure 7.1: a waiting time $w_2 = 3$ moves the original requirements (red dashed line) towards the right by 3 slots (green dot-dashed line), avoiding streaming interruptions in the first six slots (red area between the original requirements and the obtained rates, blue solid line). Since content duration can be longer than T, a non-empty buffer is required at the end of the optimization window: in particular, we require the buffer to contain the minimum between the initial amount and the remaining size of the content. In each slot j user i receives $a_{i,j}r_{i,j}$, which can be used either to satisfy the requirements in the current slot or to fill the buffer for later use. Thus we can write the following equation that describes the next buffer state:

$$b_{i,j+1} = b_{i,j} + a_{i,j}r_{i,j} - \overrightarrow{d_{i,j}} + l_{i,j} - \overrightarrow{u_{i,j}} + e_{i,j} \tag{7.3}$$

which means the buffer of user i in slot $j+1$ is obtained from the previous buffer $b_{j,i}$ by adding the received data $a_{i,j}r_{i,j}$ and subtracting the minimum requirements $\overrightarrow{d_{i,j}} - l_{i,j}$ and the extra quality $\overrightarrow{u_{i,j}} - e_{i,j}$[2]. Finally, we define $b_{i,0}$ as the initial status of the buffer of user i.

In addition, we introduce two KPIs that we will use to build the objective function for our problem. Namely, we define the fraction of continuous streaming time $\lambda_i \in [0,1]$ and the fraction of the extra quality obtained $\theta_i \in [0,1]$ as:

$$\lambda_i = \frac{1}{T}\sum_{k\in\mathcal{T}}\left(1 - l_{i,k}\overrightarrow{d'_{i,k}}\right) \tag{7.4}$$

$$\theta_i = \frac{1}{T}\sum_{k\in\mathcal{T}}\left(1 - e_{i,k}\overrightarrow{u'_{i,k}}\right), \tag{7.5}$$

where

$$\overrightarrow{d'_{i,j}} = \begin{cases} 1/\overrightarrow{d_{i,j}} & \overrightarrow{d_{i,j}} > 0 \\ 0 & \overrightarrow{d_{i,j}} = 0 \end{cases}$$

$$\overrightarrow{u'_{i,j}} = \begin{cases} 1/\overrightarrow{u_{i,j}} & \overrightarrow{u_{i,j}} > 0 \\ 0 & \overrightarrow{u_{i,j}} = 0 \end{cases}. \tag{7.6}$$

Note that when $\overrightarrow{d_{i,j}} = 0$ ($\overrightarrow{u'_{i,j}} = 0$) the interruption time $l_{i,j}$ (the extra quality outage $e_{i,j}$) is necessarily equal to 0, hence the substitutions of Eq. (7.6) are consistent.

In order to guarantee a given QoS we consider two constraints, the minimum continuous play time λ_i^* and the minimum average quality θ_i^*, defined so that $\lambda_i \geq (T - W_i)\lambda_i^*/T$ and $\theta_i \geq (T - W_i)\theta_i^*/T$. These constraints can be seen as contractual agreements that must be enforced while the content is being streamed and they change the optimization problem from a best effort resource allocation solutions where the KPIs are maximized to a joint admission control and resource allocation approach where quality of service can be guaranteed.

Finally, we build our objective function to, in order of decreasing importance, (*i*) minimize the aggregate waiting time of the system ($\sum_{k\in\mathcal{N}} W_k$), (*ii*) maximize the total continuous streaming time ($\sum_{k\in\mathcal{N}} \lambda_k$) and (*iii*) maximize the total extra quality ($\sum_{k\in\mathcal{N}} \theta_k$). Consequently, we obtain the following **MILP formulation**:

[2]Normalization between rates in a slot and amount of data is not required, because we assumed the slot length $t = 1$.

$$\begin{aligned}
\underset{A,B,L,E,W}{\text{maximize}} \quad & \sum_{k\in\mathcal{N}} (K(\lambda_k - W_k) + \theta_k) \qquad (7.7)\\
\text{subject to:} \quad & a_{i,j} \geq 0; \ \sum_{k\in\mathcal{N}} a_{k,j} \leq 1\\
& \lambda_i \geq (T - W_i)\lambda_i^*/T; \ \ \theta_i \geq (T - W_i)\theta_i^*/T\\
& l_{i,j} \geq 0; \ \ e_{i,j} \geq 0; \ \ b_{i,j} \leq b_M\\
& l_{i,j} \geq \overrightarrow{d_{i,j}} - a_{i,j}r_{i,j} - b_{i,j}\\
& e_{i,j} \geq \overrightarrow{u_{i,j}} - a_{i,j}r_{i,j} - b_{i,j} + \overrightarrow{d_{i,j}} - l_{i,j}\\
& \forall i \in \mathcal{N}; j \in \mathcal{T}\\
& \text{Eqns. } (7.3), (7.4) \text{ and } (7.5).
\end{aligned}$$

Eqns. (7.1-7.2) have been properly replaced by linear form. Note that the objective function is a linear combination of three components: $W_k \in \{0, 1, \ldots, T\}$, $\lambda_k \in [0, 1]$ and $\theta_k \in [0, 1]$, of which the first two are multiplied by $K > 1$. Since $\sum_{k\in\mathcal{N}} W_k \in \{0, \ldots, NT\}$, while $\sum_{k\in\mathcal{N}} \lambda_k \in [0, 1]$ and $\sum_{k\in\mathcal{N}} \theta_k \in [0, 1]$, the minimization of the waiting time is always addressed first in the problem. Thus, the solver assign resources so that as many users as possible obtain the required λ_i^* and θ_i^*. The weight K ensures that the solver's second priority is the continuous streaming time: ideally for $K \to \infty$ the solution would never choose quality over continuous streaming, but in practice it is sufficient to set $K \gg 1$ as $\max\{\lambda_i\} = \max\{\theta_i\} = 1$.

Having the three quantities in the objective function accommodates all possible scenarios: for instance, if the sum of the achievable rates is very large compared to the sum of requirements, the solution is likely to obtain no waiting time and continuous streaming for all users and the objective function will assign resources to maximize the extra quality. When all users need some pre-buffering, the objective function will first use resources to reduce the waiting time and then to improve the continuous streaming. The granularity of the waiting times W_i may leave unused resources between the best solution and the next, unfeasible, value of the objective function. These saved resources can be used to either improve users' λ or θ, whereas they cannot decrease the total waiting time.

7.3. Online Algorithm

A few preliminary tests showed that the MILP formulation of Eq. (7.7) is too complex (i.e. solvers need too much time) for online operations. The reasons are mainly two: MILP formulations are inherently combinatorial and the dimensionality of the problem is proportional to T^2N due to the three-dimensional matrices D and U, introduced to account for requirements shift. In this section we reduce the formulation complexity in two steps:
1) first, we decrease the problem dimensionality from T^2N to TN by replacing waiting times

with admission control: a user is admitted only if the QoS constraints can be satisfied for the whole content duration;
2) subsequently, to remove the combinatorial aspect of the MILP formulation, we approximate it with an iterative procedure based on a simpler LP approach and a binary search over a sorted list of the users.

Reduced MILP formulation: to reduce the dimensionality of the problem caused by shifting the requirement sequences according to the waiting time W_i, we introduce a binary variable s_i, representing whether a user is admitted or not in the current optimization windows: $s_i \in \{0,1\},\ i \in \mathcal{N}$, where $s_i = 1$ if user i is admitted. Users who are admitted start streaming the content immediately (i.e. $W_i = 0$) and must fulfill both QoS conditions (λ_i^* and θ_i^*) for the whole content duration. Users that are not immediately admitted can only pre-buffer data if resources are still available. We obtain the following reduced MILP formulation:

$$\begin{aligned}
\underset{A,B,L,E,S}{\text{maximize}} \quad & \sum_{k\in\mathcal{N}} \left(K(\lambda_k + s_k) + \theta_k\right) \qquad (7.8)\\
\text{subject to:} \quad & a_{i,j} \geq 0; \ \sum_{k\in\mathcal{N}} a_{k,j} \leq 1\\
& \lambda_i \geq \lambda_i^* s_i; \ \ \theta_i \geq \theta_i^* s_i\\
& l_{i,j} \geq 0; \ \ e_{i,j} \geq 0; \ \ b_{i,j} \leq b_M\\
& l_{i,j} \geq d_{i,j} - a_{i,j} r_{i,j} - b_{i,j}\\
& e_{i,j} \geq u_{i,j} - a_{i,j} r_{i,j} - b_{i,j} + d_{i,j} - l_{i,j}\\
& \forall i \in \mathcal{N}; j \in \mathcal{T}\\
& \text{Eqns. } (7.3), (7.4) \text{ and } (7.5),
\end{aligned}$$

where we replaced the shifted requirements with the original ones (Eq. (7.3-7.5) should be modified accordingly). We observe that the constraints on λ_i and θ_i are only activated if $s_i = 1$. In fact, if user i is not admitted ($s_i = 0$) the constraint becomes $\lambda_i \geq \lambda_i^* - (1 - s_i) = 0$, thus the problem accepts any value for λ_i, which means users that are not admitted can still obtain resources, but they can only pre-buffer data without playing the actual content.

In addition, the term $\lambda_k + s_k$ in the objective function has a discontinuity in $\lambda_k = \lambda_k^*$, as $\lambda_k \in [0,1]$ varies continuously, while $s_k \in \{0,1\}$ is discrete. Thus the solver will try to have as many admitted users as possible first ($\lambda_k > \lambda_k^*$). Then, after the largest set of users is admitted with guaranteed QoS, the remaining resources are distributed to either improve the QoS for already admitted users or to other users according to what requires fewer resources.

This allows us to estimate the time a non-admitted user has to wait before starting consuming the requested content:

$$W_i = T - \left\lfloor \frac{\sum_{k\in\mathcal{T}} a_{i,k} r_{i,k}}{\lambda_i^* \sum_{k\in\mathcal{T}} d_{i,k} + \theta_i^* \sum_{k\in\mathcal{T}} u_{i,k}} \right\rfloor, \qquad (7.9)$$

where the ratio between the total rate obtained $\sum_{k\in\mathcal{T}} a_{i,k} r_{i,k}$ and the needed rate to meet the requirements $\lambda_i^* \sum_{k\in\mathcal{T}} d_{i,k} + \theta_i^* \sum_{k\in\mathcal{T}} u_{i,k}$ approximates the number of slots where the content could be streamed at the agreed quality. After this time, a user is not immediately admitted into the system, but the solution is computed again to consider the impact of (*i*) requirement shift and (*ii*) prediction update.

In addition, since non-admitted users might start with a larger buffer state than new users, they will be required to maintain the same buffer state at the end of the optimization window (if the media is longer) or the remaining content size (if this is smaller than the starting buffer). Conserving the buffer between consecutive optimization windows is particularly useful when the content duration is longer than the optimization window and it is thus not possible to guarantee the QoS over its whole duration. Instead, the buffer conservation takes care of maintaining the quantity of resources that were lacking in the first round of optimization.

LP formulation: starting from the reduced MILP formulation and fixing the set of admitted users $\tilde{\mathcal{N}}$ for which $\tilde{s}_i = I(i \in \tilde{\mathcal{N}})$, a LP formulation is obtained from Eq. (7.8) setting $s_i = \tilde{s}_i$ and replacing the objective function with:

$$\underset{A,B,L,E}{\text{maximize}} \sum_{k\in\mathcal{N}} \left(K\lambda_k + \theta_k\right), \tag{7.10}$$

where $I(x)$ is the indicator function and is 1 if x is true and 0 otherwise. This formulation requires all users in $\tilde{\mathcal{N}}$ to satisfy the quality constraints. However, the set of admitted users is given as a parameter. The selection of such set is critical, since it may also lead to unfeasible problems.

Admission and Resource Control: Hereafter we propose a binary search to approximate the best feasible set of admitted users. To evaluate the set of admitted users we propose a greedy utility function to sort the users and then we define the set of admitted users of size $\tilde{N} = |\tilde{\mathcal{N}}|$ as the set composed of the first $\tilde{N}$ users. By means of a binary search over the size of the admitted set $\tilde{N}$, we find the largest size $\tilde{N}$ for which the problem of Eq. (7.10) is feasible.

The sorting function has to weight how efficiently resources are used to satisfy users' requirements. This efficiency depends on almost all the input parameters of our problem and, in particular, it is related to the sequence of achievable rates: high rates in the early slots allow a user to fill its buffer and avoid to use low rates slots, but a high rate in a slot where many users have high rates means that many users will try to use resources in the same slots.

Since evaluating all these parameters for every combination of users would be as complex as solving the original problem, we follow an indirect approach: we compute the schedule that maximizes $\sum_{k\in\mathcal{N}} (K\lambda_k + \theta_k)$ if no QoS is enforced ($\tilde{\mathcal{N}} = \emptyset$). In such a case, no user is required to meet any condition on the QoS and resources are assigned, first, to maximize the overall continuous streaming time and, then, the average quality. Thus, the solution of Eq. (7.10) is certainly feasible and obtains the resource allocation $\tilde{A}$.

Algorithm 7 Admission and Resource Control

```
Input: $R, D, U, b_M, \lambda_i^*, \theta_i^*$.
Output: $\tilde{A}, \tilde{\mathcal{N}}$
  $N_{\min} = 0$, $N_{\max} = N$
  Compute $A$, from Eq. (7.10) with $\tilde{N} = N_{\max}$
  if Problem feasible then
      $\tilde{a_{i,j}} = a_{i,j}, \tilde{\mathcal{N}} = \mathcal{N}$
  else
      Compute $\tilde{A}$, $\tilde{\lambda}_i$, $\tilde{\theta}_i$, from Eq. (7.10) with $\tilde{\mathcal{N}} = \emptyset$
      Compute $\phi_i$ from Eq. (7.11) $\forall i \in \mathcal{N}$
      Sort $\mathcal{N}$ in descending order of $\phi_i$
      while $(N_{\max} - N_{\min}) > 1$ do
          $\tilde{N} = (N_{\max} + N_{\min})/2$
          Compute $A$, from Eq. (7.10) with $\tilde{\mathcal{N}} = \{i \in \mathcal{N} | i \leq \lfloor \tilde{N} \rfloor\}$
          if Problem feasible then
              $N_{\min} = \tilde{N}$
          else
              $N_{\max} = \tilde{N}$
          end if
      end while
      $\tilde{a_{i,j}} = a_{i,j}$
  end if
```

According to the scheduling $\tilde{A}$, each user i is characterized by the two KPIs $\tilde{\lambda}_i$ and $\tilde{\theta}_i$. Consequently, the least efficient user i is the one that has the lowest $\tilde{\lambda}_i$. In case of equal $\tilde{\lambda}_i$ we choose over $\tilde{\theta}_i$. In case of both equal $\tilde{\lambda}_i$ and $\tilde{\theta}_i$, we consider the amount of used resources. Therefore, we propose the following sorting function:

$$\phi_i = \frac{T(K\tilde{\lambda}_i + \tilde{\theta}_i)}{\sum_{k \in \mathcal{T}} \tilde{a}_{i,k}}, \tag{7.11}$$

where $\sum_{k \in \mathcal{T}} \tilde{a}_{i,k}/T$ is the total fraction of resources used.

Once that the sorting function has been defined, we can apply a binary search over the size of the set of admitted users. We call the algorithm Admission and Resource Control and its pseudocode is given in Algorithm 7. The convergence of the binary search is ensured by the sorting of the users: in fact any given set $\tilde{\mathcal{N}}$ always includes all the elements of the smaller sets, thus, if it makes the problem unfeasible, no larger sets can be feasible.

In what follows we provide a few **practical considerations** about its realization in cellular networks. With reference to current LTE, Fig. 7.3 shows a high level diagram of an eNodeB where only the relevant functionalities are drawn. The prediction and context information functionalities are drawn outside the eNodeB as they contain network wide information that are not specific to any eNodeB. However, it is possible to cache locally in the eNodeB the information that is more frequently used. Also, while the mobility prediction may be computed outside the eNodeB, the

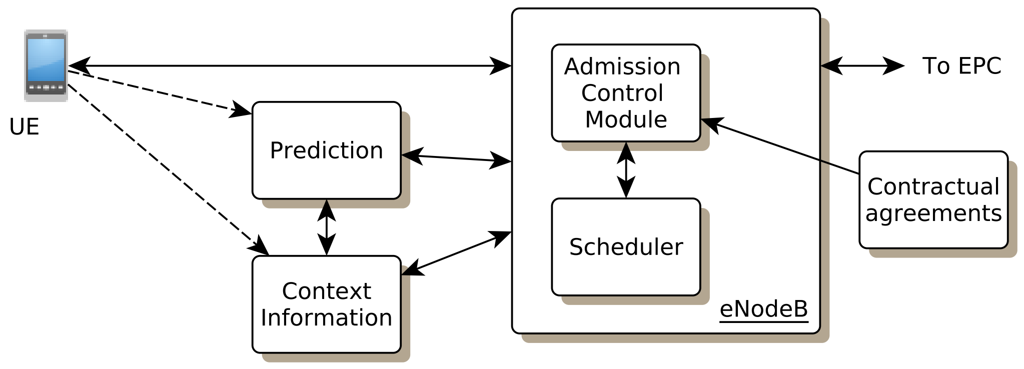


Figure 7.3: eNodeB high level diagram highlighting the relationship among the different modules.

short term achievable rate variation might be computed internally as well. The input parameters of the problem ($r_{i,j}$, $d_{i,j}$, ui, j) are obtained by combining prediction, context information and admission control functionalities. The contractual agreement function governs the constraints of the problem and defines λ_i^* and θ_i^* for all users.

The admission control function is placed in parallel to the scheduler in order for the former to provide input to the latter without changing the main scheduling logic. These two functions operate at different time granularity: while the scheduler makes decisions every few milliseconds, the admission control time slots are in the order of seconds. The admission control should be able to modulate the user weights used by the scheduler. This allows the system to enforce admission control indirectly: the weight of a user which is not admitted in the current admission time slot is set to zero, while admitted users receive weights proportional to the fraction of resources assigned by the admission control.

In practice, whenever the admission control solution is re-evaluated, the admitted status of users that still have to complete their stream should be preserved. This can be achieved using an additional equality constraint requiring s_i to be larger or equal than the value obtained in the previous evaluation. New user arrivals can be managed either synchronously if the admission control time slots are smaller than 1 second or asynchronously if longer. In this last case, the users already admitted must preserve their condition.

7.4. Simulation Results

This section presents the results of our evaluation campaign, which can be grouped in three parts: (*i*) the first part analyzes the computational complexity; (*ii*) the second evaluates how far the solution obtained by our approximation is from the original problem; (*iii*) the third part discusses the benefits of the combined admission control and resource allocation technique with respect to the baseline solution and an anticipatory technique that does not enforce QoS.

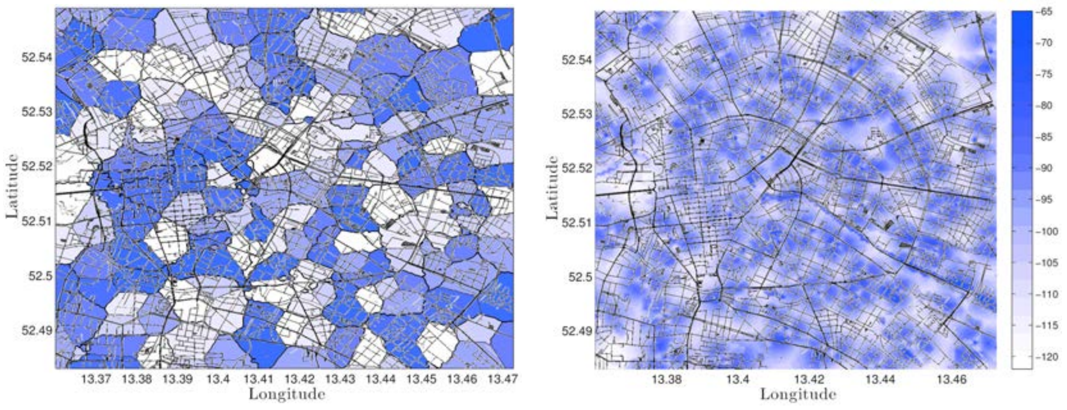


Figure 7.4: Coverage and pathloss maps of Berlin as measured by the MOMENTUM project.

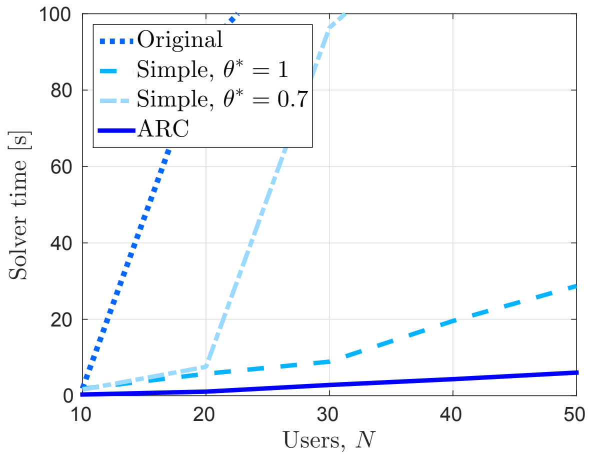
In particular we consider the following problems:

- *Original*: problem formulation of Eq. (7.7),
- *Simple*: mixed integer linear formulation of Eq. (7.8),
- *Admission and Resource Control (ARC)*: online iterative approach of Algorithm 7,
- *Resource Allocation (RA)*: anticipatory resource allocation without QoS (e.g. [7]),
- *Baseline*: plain proportionally fair scheduling.

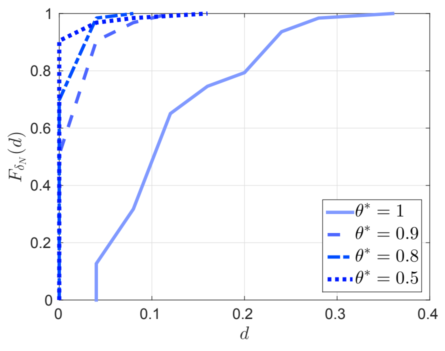
Our evaluation campaign considers an LTE network scenario based on the pathloss data provided by the MOMENTUM project [160]. For each evaluation round we generate a random mobility trace in a 12×6 square kilometer area of Berlin (centered at latitude $52.52°$ North and longitude $13.42°$ East). Fig. 7.4 shows a map of the cell topology (left) in the considered area. From the mobility trace, we generate a pathloss trace computed on the pathloss map (right). Finally, we account for fast fading as in the model discussed in [197] to obtain the achievable rates and we averaged results over 200 repetitions of 5-minute scenarios.

The requirement traces are constant and equal for all the users to simplify the discussions of the results. However, all the formulations support any type of requirements. In particular, we set $d_{i,j} = 0.4$ Mbps and $u_{i,j} = 4.6$ Mbps to represent the different qualities available for video streams of resolution ranging from 360p (400 Kbps) to 1080p (< 5 Mbps). Unless specified otherwise, $\lambda_i^* = \lambda^* = 1$ for all users. This means that in all the following results it is required for the streaming to have no interruption. To prioritize continuous streaming time over extra quality we chase $K = 100TN$ for all the simulations.

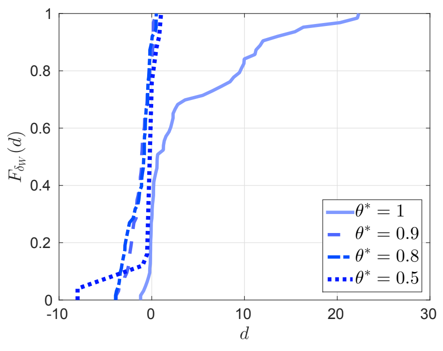
The first tests aim to understanding which of the three formulations can be used to implement a real-time admission control and resource allocation mechanism based on system state prediction. The main challenge of such a module is to obtain a solution within the validity time of the



(a) Complexity



(b) eCDF of δ_N



(c) eCDF of δ_W

Figure 7.5: Evaluation of the computational time and the optimality of the different approaches.

prediction. To this end, we evaluate the three formulations over repeated instances with varying problem size, i.e., number of optimization variables involved in the specific instance.

Eq. (7.7) has dimensionality proportional to T^2N, while the simpler formulation of Eq. (7.8) has a size proportional to TN. However both include integer variable, while Algorithm 7 consists of at most $\log_2 N$ iterations of a simple LP program of size proportional to TN.

In our evaluation we explore the following parameters: users $N \in [10, 50]$, slots $T \in [10, 50]$, quality requirements $\theta_i^* = \theta^* \in [0.5, 1], \forall i$ and we compare the average computational time[3] obtained by the three formulations using GUROBI [205]. In Fig. 7.5(a) we fix the number of slots $T = 30$ and we plot a solid curve for ARC, dashed ($\theta^* = 1$) and dot-dashed ($\theta^* = 0.7$) curves for Simple and a dotted curve for the Original approach for $N = [10, 50]$[4]. We do not plot curves for different θ^* for the original and ARC formulation as this parameter has minimal impact on the computation time. Instead, we plot two curves for the simple formulation for $\theta^* = 1$ and $\theta^* = 0.7$, because we observe that if the system does not require the full quality to be delivered, the resource allocation has more degree of freedom and decreases the solution

[3]In all cases we stop the computation after 100 seconds.

[4]We do not report the curves obtained for a fixed N varying the number of slots, because they show a similar trend.

speed. The original formulation becomes too slow very rapidly, while the simple formulation can be computed in less than 10 seconds if $\theta^* = 1$. However, for lower θ^* the simple formulation is affordable for very small problem instances only. This is due to the fact that for small problem instances the solution becomes trivial as almost all users can be admitted. Finally, ARC obtains a solution in an affordable time for all the problem sizes.

In the second set of results we compare the solutions obtained by the simple MILP and the ARC approaches. In particular, we evaluate the number of admitted users $\hat{N}$ (MILP) and $\tilde{N}$ (ARC) and the average waiting time $\hat{W} = \sum_{k\in\mathcal{N}} \hat{w}_k(N - \hat{N})$ (MILP) and $\tilde{W} = \sum_{k\in\mathcal{N}} \tilde{w}_k/(N - \tilde{N})$ (ARC) computed with Eq. (7.9). We choose $N = 25$ and $T = 50$ and we vary $\theta^* \in \{1, 0.9, 0.8, 0.5\}$. Finally, for each repetition we compute $\delta_N = (\hat{N} - \tilde{N})/N$ and $\delta_W = \hat{W} - \tilde{W}$.

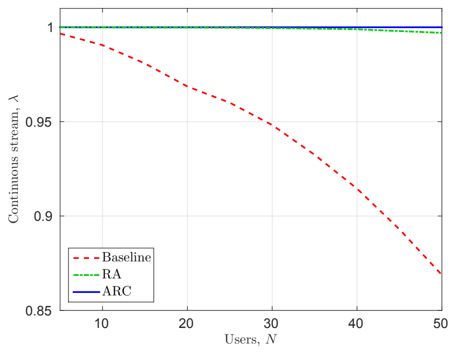
Fig. 7.5(b) and Fig. 7.5(c) plot the empirical Cumulative Distribution Function (eCDF) of δ_N and δ_W respectively. Different constraints $\theta^* \in \{1, 0.9, 0.8, 0.5\}$ are plotted with solid, dashed, dash-dotted and dotted lines respectively. The former figure illustrates that the ARC approach closely approximates the number of admitted users with respect to the MILP formulation for all but $\theta^* = 1$. In this case, the exact solution of the problem requires the maximum quality to be delivered in every slot to admit a user. Thus, the approximate formulation is less likely to find the exact combination of users. Similarly, Fig. 7.5(c) shows that for the average waiting time ARC obtains a good approximation. While in the previous figure the domain of the eCDF was limited to positive values, here δ_W can assume negative values, too: in fact, by admitting less user in the system, more resources remains for the non-scheduled users that can start the streaming earlier.

The final set of results compares Baseline (red dashed line), RA (green dash-dotted line) and ARC (solid lines from darker to lighter shade of blue representing $\theta^* \in \{1, 0.9, 0.7, 0.4\}$) to investigate the improvements offered by our proposal over existing solutions. The results for RA is obtained using the formulation of Eq. (7.10) with no admitted users, hence no QoS is enforced. In this set of graphs we vary both $N \in [5, 50]$ and $\theta^* \in [0.1, 1]$.

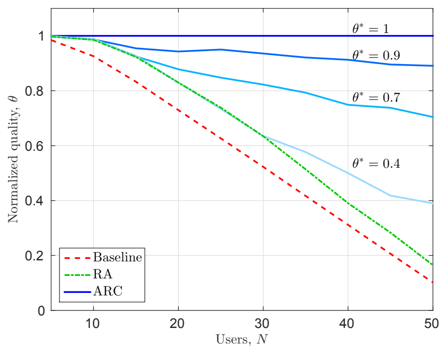
Fig. 7.6(a) shows the average fraction of continuous streaming obtained by the three approaches. Baseline does not leverage prediction cannot avoid streaming interruption. As the number of users increases, the average interruption time reaches 15%. Both RA and ARC show almost no interruptions for any user. They only differ if $N > 30$ for which ARC drops a few users to enforce QoS.

Fig. 7.6(b) shows the average fraction of obtained quality (1 means that all the streams obtain the maximum quality in every slot) for the three approaches. The overall quality obtained decreases with the number of users for all approaches to different degrees. RA and ARC always deliver higher quality than Baseline. In addition, we plot 4 curves for different quality constraints for ARC. The two predictive approaches, ARC and RA obtain the same quality as long as the number of users is small enough to sustain the required QoS, then RA starts violating the constraint, while ARC reduce the set of admitted users.

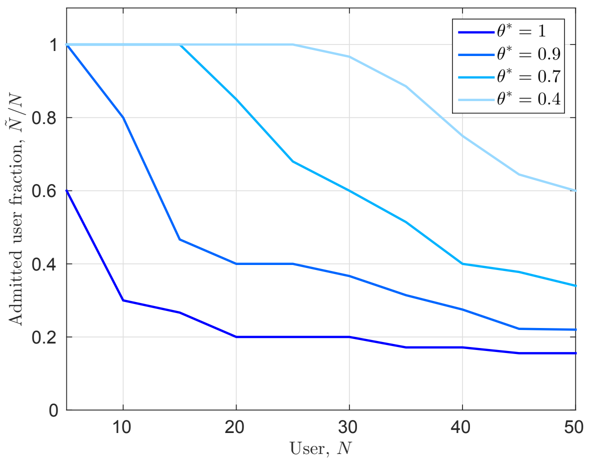
Finally, Fig. 7.6(c) shows the average fraction of admitted users $\tilde{N}/N$ for ARC. The com-



(a) Continuous streaming



(b) Extra quality



(c) Admitted users

Figure 7.6: Evaluation of the performance of the joint admission control and resource allocation solution.

parison between the last three figures highlights the tradeoff intrinsic of our solution: the joint admission control and resource allocation is able to tradeoff the number of admitted users and the guaranteed QoS. For instance, to obtain a stream with no interruption at 40% of the maximum quality, only 30 of the 50 requesting users can be admitted at once.

Part III :   Tools and Practical Challenges

Summary

In this part we describe the tools and methodologies developed and used to validate the theoretical results presented in the previous part of this thesis. In particular, we considered two main problems: 1) What is the performance of anticipatory networking solutions when applied to real world (i.e. not synthetic) scenarios? 2) Can mobile phone applications obtain precise and accurate estimates of their achievable link data rate to use to drive predictive optimization techniques?

In Chapter 8, we present the main tool designed and developed to address both problems: a decoder of the downlink control channel of LTE. Decoding this information provides access to the complete scheduling data of an LTE base station, such as, users temporary identifiers, assigned resource blocks, and modulation and coding schemes. All this information is provided for every transmission, thus, every millisecond. The tool is freely available at: `https://git.networks.imdea.org/nicola_bui/imdeaowl`.

Chapter 9 evaluates how accurately can mobile phone applications estimate the achievable link data rate by providing link layer measurements obtained with my sniffer. We find out that accurate and precise measurements can be obtained with mobile phones even in case of short-lived communications (e.g. 50 ms or 100 KBytes). However, different phones present different biases.

Chapter 8

OWL: a Reliable Online Watcher for LTE Control Channel Measurements

In this chapter, we introduce the Online Watcher for LTE: a sniffer capable of reliably decoding LTE control channel information. Our tool[1] is meant for researchers and SMEs that need a simple and economic solution to perform reliable measurements on LTE physical communications between mobile phones and base station. OWL is built on top of srs-LTE [214] that provided us with modular and efficient implementations of LTE physical channels and basic procedures and works with a few Software Defined Radios (SDRs), such as bladeRF [215] and USRP [216], capable of LTE signal sampling. In particular, we extended srs-LTE by implementing an online procedure to decode all Downlink Control Information (DCI) transmitted on the Physical Downlink Control Channel (PDCCH) [217]. Our solution is more efficient than previous attempts, because we are able to collect and maintain a list of active Radio Network Temporary Identifier (RNTI)s, which identify UEs within a given cell (eNodeB). In fact, RNTIs are the key for mobile phones to distinguish the control messages destined to them and to verify the success of DCI decoding. This technique provides OWL with two very desirable features: 1) it is very reliable as it can be verified via the Cyclic Redundancy Check (CRC) field and 2) it can be executed online on inexpensive hardware, since it does not need heavy computation. We measure OWL's reliability by comparing the schedule information obtained from DCIs to the used network resources by means of power measurements on the raw signal: in more than 99% of the captured frames in our tests OWL detects all the scheduled transmission, scoring an average 99.85% successful decoding ratio overall. Therefore, OWL can be used as a ground truth check for mobile phone measurements, to perform extensive mobile networks measurement campaigns or to evaluate mobile networks performance and functionalities.

The rest of the chapter provides the related work in Section 8.1, the basic LTE details in Section 8.2, the description of OWL and its architecture in Section 8.3, and OWL's performance evaluation in Section 8.4.

[1]The code is available at: `https://git.networks.imdea.org/nicola_bui/imdeaowl`.

8.1. Related work

To the best of our knowledge the first non-commercial attempt to decode LTE control information has been LTEye [218]: DCI messages are not encrypted, but only the intended receiver can verify the successful decoding, because the CRC field of the message is scrambled (binary exclusive OR operation) with the UE's RNTI. To decode DCIs without knowledge of the destination RNTI, LTEye, first, assumes the decoding to be successful, then obtain the destination RNTI from the CRC field of the message XORed with the CRC computed on the decoded data. The shortcoming of this is that the CRC cannot to be used to validate the decoding operation. To solve this, the authors propose to re-encode the decoded message and to compare the result with the original bits received before the decoding operation. Although feasible under almost bit-perfect channel condition, this approach suffers from low reliability as has been verified in [219].

The latter paper proposes RMon, another technique to monitor the resource allocation on the Physical Downlink Shared Channel (PDSCH): by comparing the received signal strength to LTE reference signals [220], RMon is able to evaluate which resource block is used regardless of the control information. Although quite reliable, this approach does not allow to obtain any additional information beyond the fraction of used resources. A very recent solution is designed by Falkenberg et al. [221] to estimate mobile phone connectivity.

Instead, thanks to the list of active RNTIs, OWL is both reliable, because it can verify the DCI decoding with the CRC, and expressive, since it can access all DCI fields. Of course, commercial products might offer similar features albeit at a much higher price and complexity, e.g., QXDM [222], Actix Analyzer [223], or TEMS investigation [224].

For what concerns open-source LTE implementations, we use srs-LTE [214] for its very efficient implementation. In addition, the modularity of the architecture and the adherence to the standard terminology allowed us to realize OWL starting from the provided example program to record and synchronize the LTE signal. Alternative approaches include gr-LTE [225] a solution based on GNU Radio, and openLTE [226], which is more focused on the actual transmission and reception of PDSCH and is more suitable for isolated experiments where both UEs and eNodeB are controllable.

8.2. Control Channel Decoding

This section is a mini-guide to LTE physical channels and procedures needed to understand the operations performed during the control channel decoding. In particular, we cover synchronization procedures and the related channels, RNTI types and the random access procedure and, finally, the control channel and the information carried by DCI messages. In what follows, we limit our description to frequency-division duplex and standard cyclic-prefix duration and most of LTE's subtleties are omitted due to size limitation of the paper. The interested reader is referred to [220] for other details of LTE.

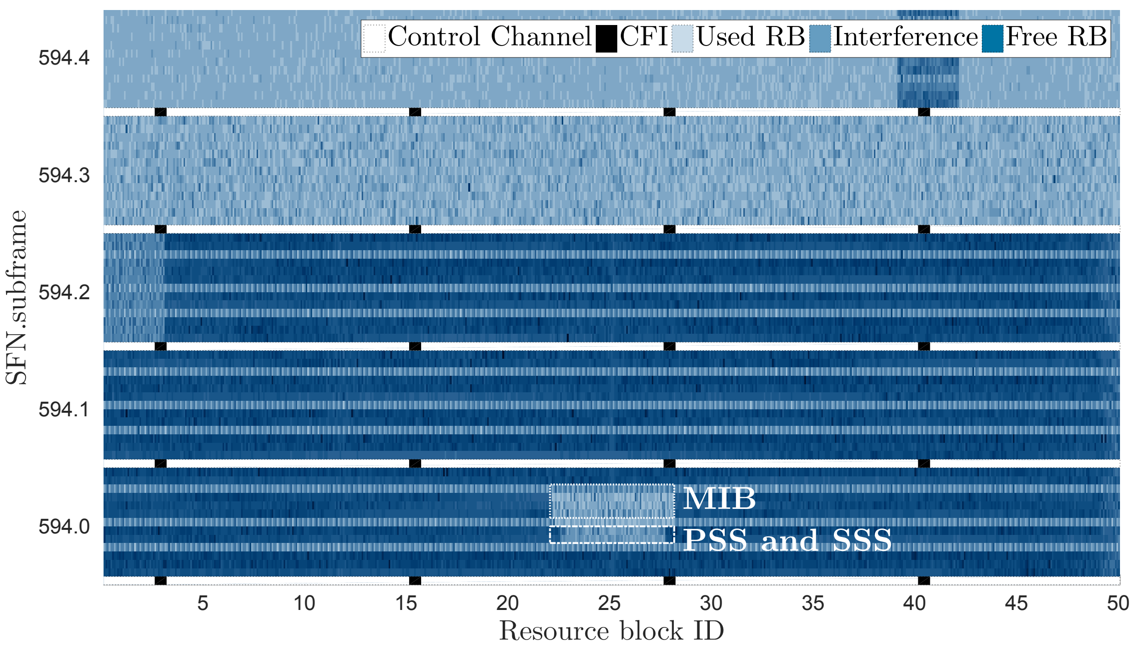


Figure 8.1: Annotated capture of half a frame of a 10 MHz LTE signal. The OFDM grid spans resource blocks on the x-axis and subframes on the y-axis. We highlighted the synchronization sequences (PSS and SSS) and the MIB in the center of the band. The horizontal lines representing the control channel are highlighted in white, while the CFI elements within the control channel are drawn in black.

Figure 8.1 is an annotated power chart of half a frame of a 10 MHz LTE signal. It is obtained by expanding the OFDM grid in 600 sub-carriers (x-axis) and 70 symbols (y-axis). A Resource Element (RE) is the minimum two-dimensional unit (1 sub-carrier $\times$ 1 symbol), a RB consists of 84 REs organized over 7 symbols and 12 subcarriers, 7 symbols form a slot, 2 slots form a subframe and 10 subframes are a 10 ms frame.

In order to synchronize with a given eNodeB, the user equipment (UE) computes the correlation between the received signal and three known Zadoff-Chu sequences. This allows the UE to acquire the location of the PSS and to decode the SSS. Both can be found in subframes 0 and 5 in every frame. By doing so, the UE can compute the eNodeB Physical Cell ID (PCI) and the system timing, which are needed to identify all the remaining physical channels in LTE. The next synchronization step is decoding the MIB, which is located in subframe 0 of every frame and carries the System Frame Number (SFN) as well as other system parameters.

RNTIs are 16-bit identifiers used by the eNodeB to distinguish among the many UEs connected at any given time. Among the different types of RNTI, only two are relevant to our procedures: random access RNTI (Random Access RNTI (RA-RNTI)) and Cell RNTI (C-RNTI). The former only takes values in $[1 - 10]$ and is used during the random access procedure to allow the eNodeB to address an unknown UE. The latter can take any unreserved value in $[0x003\text{D}-\text{FFF3}]$ and is assigned to the eNodeB at the end of the random access procedure. A brief overview of the random access procedure is as follows: 1) the UE sends one out of 64 possible preambles

(Zadoff-Chu sequences) in subframe i; 2) the eNodeB sends a Random Access Response (RAR) message in which a temporary C-RNTI is assigned to the UE; 3) the UE sends a Radio Resource Control (RRC) connection request message; 4) the eNodeB responds with contention resolution message to UE. In order for the UE to receive the RAR, the related DCI is sent to the RA-RNTI address $i + 1$, which is defined by the subframe where the UE sent the preamble. The C-RNTI received during step 2 is only confirmed in step 4; in fact, if two or more UEs selects the same subframe for sending the preamble, all of them receives the RAR with the same information. However, only one of them will successfully complete step 3, thus, receiving the final confirmation from the eNodeB. In any case, the temporary C-RNTI sent in the RAR is assigned to one of the users participating in the random access procedure.

Note that the DCI sent to the RA-RNTI only carries information for the UE to decode the RAR, but the actual RAR is a proper RRC message sent in the shared downlink channel. Thus, the UE can decode the 6 bytes of the actual message, which consists of a short header, the time alignment, the upload grant to let the UE send the message in step 3 and, in the last 2 bytes, the C-RNTI that is going to be used by the user winning the contention.

LTE schedule is completely governed by the eNodeB and no communication can happen without an explicit control message being issued on the control channel that occupies the first symbol(s) of each subframe. In the figure, we colored all control channel symbols in white for an easier identification, whereas the remaining REs are colored in different shades of blue (light, dark and medium for used, free and interfering RBs). The actual number of symbols used for the control channel is specified in the Control Format Indicator (CFI) a 32-bit sequence spanning 16 RE, the position of which depends on the PCI (in black within the control channel in Figure 8.1) and that can assume a value in $\{1, 2, 3\}$. Depending on the size of the control channel and the system bandwidth, UEs need to monitor different locations on the control channel, since, to avoid collisions, a control message destined to a given RNTI can only occupy a subset of the available locations.

Due to LTE's flexibility and its many revisions, there exist many different DCI formats. However, here we only provide the common characteristics that allows OWL to monitor the cell traffic. First of all, every DCI format specifies whether it is related to the uplink or the downlink: this information is either derived by the size of the message, if it is unique for a given format, or by the first bit of the message, otherwise. The second field which is always present in transmission related DCIs is the MCS field: 5 bits that specify the modulation and the code rate that will be used in the corresponding transmission. The last two pieces of information that OWL extracts from DCIs are the number of used resource blocks N_{RB} and the transport block size. The definition of the former depends on the actual DCI format, while the latter is derived by using MCS and N_{RB} as indices in a lookup table. The complete definitions can be found in [227]. Finally, DCI messages have a CRC footer, which is the result of a XOR operation between the actual CRC computed over the message payload and the C-RNTI of the destination UE.

8.3. OWL Architecture

The OWL software architecture is composed of three processes: 1) a *synchronized signal recorder*, 2) the actual *control channel decoder*, and 3) a *fine-tuner* that is used when a control message is expected to be found on the control channel, but the main process cannot decode it. Finally, we develop an auxiliary *verifier* tool that checks whether the decoded DCIs match the actual resource allocation on the PDSCH.

8.3.1. Synchronized signal recorder

OWL's signal recorder inherits most of its functionalities from the synchronized signal recorder provided by srs-LTE. This tool, first, synchronizes the software to the eNodeB transmissions by means of PSS and SSS correlation, then acquires the remaining information by decoding the MIB, and finally it writes an output file starting from the first symbol of the first frames for which it obtained a successful MIB decoding. However, it might happen that the system synchronization degrades without the recorder being able to notice, in particular for recordings longer than a few seconds.

To improve this, we provide the recorder with a synchronization check at the beginning of every frame. In addition, we provide the recorded trace with an error log that tracks any synchronization issues and any other software related error that might hamper the following operations.

8.3.2. Control channel decoder

OWL's main component is the control channel decoder. It can work either online while the signal is being sampled by the SDR or offline processing prerecorded traces. Our control channel decoder inherits from srs-LTE the basic decoding functions, such as CFI decoding, channel equalization and mapping. However, srs-LTE provides all the functionalities as they would have been implemented in a UE. Instead, OWL needs these functions to be extended to cover all possible control channel allocation: in particular, while a single UE can monitor a limited set of control channel locations, OWL needs to extend the procedure to all possible locations and DCI formats.

In any case, both srs-LTE and OWL only perform actual DCI decoding if there is an ongoing transmission on the REs of the scanned location. If this is the case, the decoding procedure is repeated for all possible DCI sizes. srs-LTE considers the decoding operation successful if the CRC field, scrambled with the CRC computed on the data, matches the C-RNTI of the UE under test. Instead, OWL only requires that any of the C-RNTI of the active list matches with the decoded message.

Since the C-RNTI list is empty when the system starts, OWL needs to populate it while decoding the control channel. To do so, OWL can either 1) exploit the random access procedure or 2) verify the decoding success by re-encoding the DCI as LTEye does. In the former procedure, whenever a DCI is decoded with the CRC field XORed with a RA-RNTI ($[1 - 10]$), not only is it considered a successful decoding, but also the RAR message, which is sent in PDSCH at the RBs specified in the DCI by means of MCS and N_{RB}, is actually demodulated and decoded and provides OWL with a new C-RNTI to be inserted in the active list.

LTE RRC messages are coded using ASN.1 [228], but the particular configuration of the RAR messages allow us to simplify the decoding by just taking the last two bytes of the message, because the C-RNTI is always specified in this location. In addition, since the actual RAR message is provided with a CRC field, OWL is able to evaluate the correctness of the whole operation by verifying the message checksum against the CRC field.

Also, OWL implements LTEye re-encoding procedure to bootstrap the list for those C-RNTIs that were assigned before the logging started and to recover from the missed random access procedures in the unlikely event of de-synchroni-zation. This gives us the added benefit to be able to compare the two methodologies: whenever a transmission is detected on the control channel we verify it both by checking the re-encoded message against the received symbols and by checking whether the C-RNTI is in the active list.

C-RNTIs are just temporary identifiers and, after a complete SFN cycle (10.24 seconds) of inactivity, a UE needs to perform the access procedure again to obtain a new one. For this reason, OWL resets all the RNTIs in the list that are inactive for more than a SFN cycle. Finally, while OWL uses the LTEye re-encoding procedure to bootstrap the RNTI list, at steady state we verified that OWL effectively detects all new RNTIs assigned by the eNodeB. As such, we only enable the DCI re-encoding when OWL detects a DCI message whose CRC is not XORed with a C-RNTI in the active list. This makes OWL both robust, because of the actual decoding verification, and computationally effective, because unneeded re-encoding operations are avoided.

We evaluate the offline control channel decoder performance and, on a single Core i3 processor, the overall computational time is about half the length of the recorded trace. Similarly, the online decoder works without ever interfering with data stream arriving from the SDR.

8.3.3. Fine-tuner

While, theoretically, the control channel decoder should be able to decode all DCIs, we identify a few rare conditions for which power is detected on the control channel, but no DCI message has been decoded. We believe that these conditions are due to either equalization or synchronization problems. The fine-tuner is able to correct the majority of these issues by iteratively performing the decoding operation on the specific location only and varying the timing offset of the LTE signal.

The drawback of the fine-tuner is that its operation time is proportional to the number of uncertain control channel locations. Our tests show that the fine-tuner takes less than the trace duration in 50% of the cases, less than five times the trace duration in 90% and up to ten times in the remaining 10%. However, they also show that the fraction of DCI message fixed by the fine-tuner is always lower than 5% of the overall decoded messages and the actual fraction is independent of the time taken to decode it; in fact the time only depends on the number of uncertain locations.

8.3.4. Pipeline

The overall OWL solution coordinates as many parallel processes as cores are available on the CPU denoted by k. The first process continuously records the LTE signal and cyclically switches the saving location among k files. These files are located on ramdisks in order not to interfere with the trace recording itself. As soon as the first process switches to the next saving location, the second process runs the control channel decoder on the recorded trace. This process produces the main output and identifies whether and where there are uncertain locations in the control channel trace. As soon as the second process is done, a new process is started to run the fine-tuner on the trace. In case the fine-tuner takes longer than the time by which the first process needs again the file to save the next trace, we force the fine-tuner to timeout before this happens. In this way OWL might lose a few control messages, but does not stop the trace recording. Also, the fine-tuner processing can last at least $k - 2$ times the length of the recorded trace and OWL hardware can be chosen to reduce the likelihood for this to happen to a minimum.

8.3.5. Verifier

Finally, to verify whether the decoded information matches the actual PDSCH resource allocation, we develop a simple tool that takes as inputs the decoding log and the raw LTE signal trace. For each subframes it computes how many RBs are detected by OWL by summing all the N_{RB} values of downlink messages. Similarly, it evaluates for each subframe and for each RB whether the average power measured on the PDSCH is higher or lower than the power measured on the reference signals that are the closest to the related RB. While the control channel decoder can decode both uplink and downlink schedule, the verifier tool can only measure downlink information with a single SDR, because in Frequency Division Multiplexing (FDD) systems the uplink physical channel is separated from the downlink by a few hundred MHz. Hence, in this paper we can only systematically verify the downlink schedule and we leave the development of a verifier tool for the uplink channel using two SDR for future work.

Figure 8.2 visualizes the result of the power analysis of the verifier tool performed on the same frame used in Figure 8.1. By comparing the two figures, it can be seen that the power analysis can effectively identify ongoing transmission (lighter areas of Figure 8.1 correspond to taller bars in Figure 8.2). Also, the first RBs of subframe 594.2 are correctly identified as interference.

8.3.6. OWL release details

OWL extends srs-LTE by adding the following:

- support for DCI formats 1B, 1C, 1D, 2, 2A
- automatic decode of DCIs sent to RA-RNTIs
- random access response messages decoding

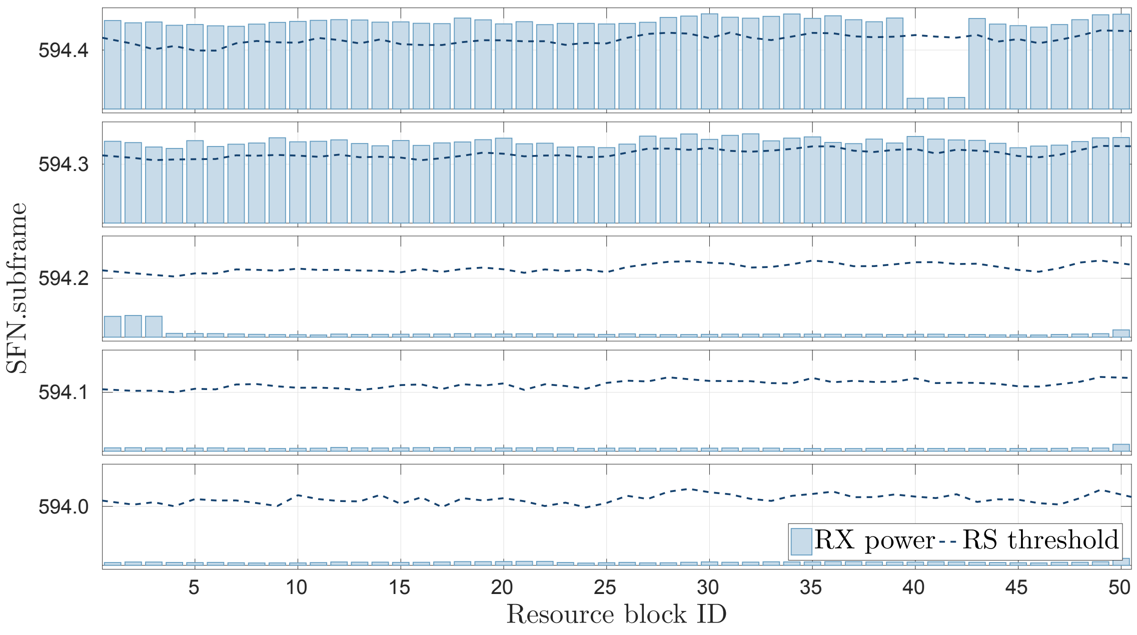


Figure 8.2: Results of the verifier on the same locations of Figure 8.1: the five rows compare the reference signal threshold to the average power measured on the RBs.

- C-RNTI list management
- DCI verification by re-encoding.

In order to run OWL, the SDR must support a LTE compatible sampling rate: 30.72 Msps (samples per second) to be standard compliant, but we successfully tested OWL at 23.04 Msps for 20 MHz bandwidth and 11.52 Msps for 10 MHz. The PC must be able to receive the recorded stream from the SDR and store it: this can be achieved by means of USB3, 1Gbit Ethernet (up to 10 MHz only) and 10Gbit Ethernet; although we did not try less capable devices, we successfully used OWL on Core i3 PCs by temporarily storing and decoding the traces in RAM and only using the physical disk to log DCI information.

At the moment of writing this paper, OWL's alpha release is already available at `https://git.networks.imdea.org/nicola_bui/imdeaowl` and is being tested by a small group of colleagues. We currently plan to run the alpha testing until the end of September and to release a fully documented beta version by the conference date in the same repository. OWL is completely open-source and it is released under the Affero General Public License v3.

8.4. Results

In this section we provide two sets of results: a first set validates OWL and compares it to LTEye, while second provides an example of analysis realized with our tool. All the tests of this section are performed by capturing a 10 MHz LTE channel in the frequency band at 1854.1 MHz

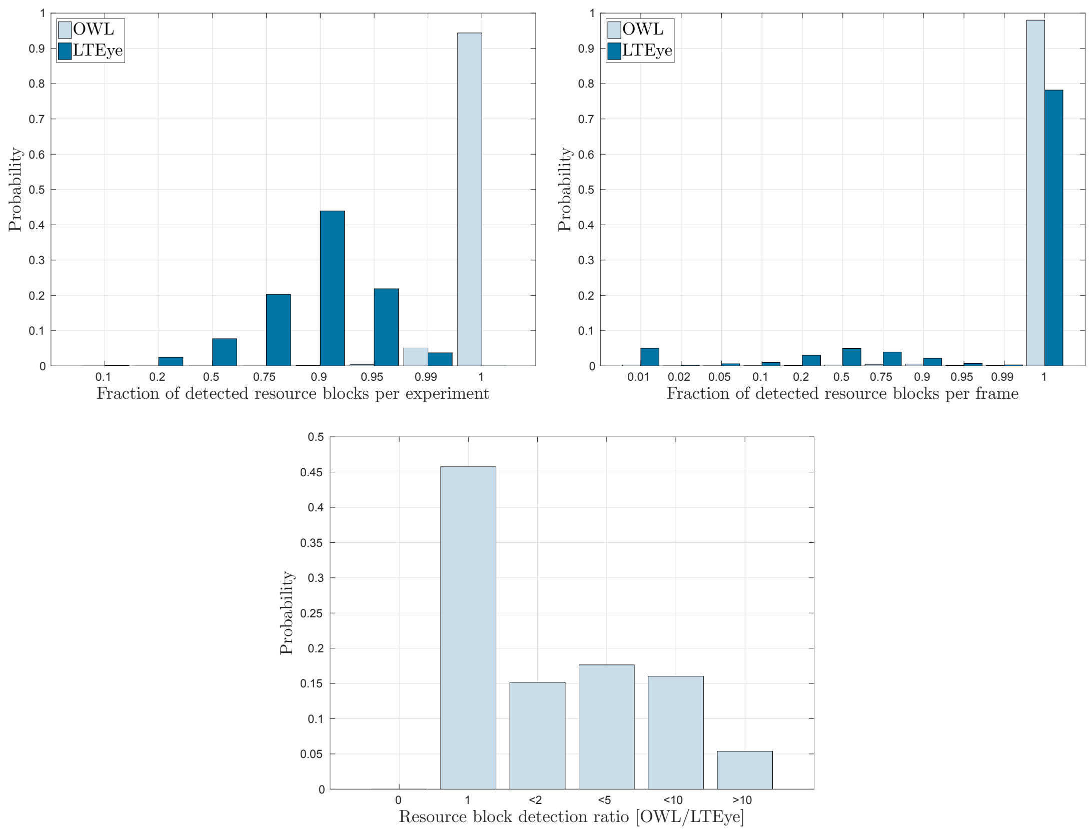


Figure 8.3: OWL validation campaign results. On the left plot we show the likelihood (y-axis) of detecting the fraction of RBs specified on the x-axis computed per frame; the central plot is similar, but measured per frame; the plot on the right illustrates the ratio between the number of decoded RBs of OWL and LTEye.

through a BladeRF x40 SDR connected to 4-processor Core i3 mini PC equipped with 4 GB of RAM and running Ubuntu 14.04.

In order to compare OWL and LTEye we run more than a thousand experiments in which we recorded 5-second traces that we subsequently decoded with OWL. In all experiments we let the fine-tuner process end, to obtain the maximum number of decoded messages.

To evaluate the performance of LTEye, after each DCI decoding we verified its success by re-encoding the message and comparing it to the received signal. If the two differ for less then 2% of the bits we count the message as a valid decoding for LTEye. Note that, for the sake of fairness we compute this after having processed the trace with the fine-tuner in order to compare OWL's procedure based on random access to LTEye re-encoding solution. Also, we choose the test location in order to have the best possible reception in our space from a nearby eNodeB.

Finally, we compute the number of RBs effectively used in PDSCH by running the verifier tool on the raw captures. Figure 8.3 (left) evaluates the fraction of RBs detected by OWL and LTEye compared to those detected by the verifier in each frame. We group the results in bars that show in the ordinate the probability to successfully decode a given fraction of RBs (x-axis) for the two solutions. In all figures the x-axis is modified in order to highlight where the probability distributions concentrate. OWL decodes all the RBs in about 95% of the tests and in the remaining 5% only miss 1% of them. Conversely, the most frequent result for LTEye is decoding 90% of the RBs and it never successfully decode all RBs in a test.

The central plot shows a similar result, but, instead of evaluating the detection ratio averaged over experiments, it plots the results for each frame: OWL successfully decode all RBs in almost 99% of the frames, while LTEye achieves less than 80%. Both solutions have non-zero probability for all detection ratios, since different frames might have a variable number of allocated RBs and a single error may represent different detection ratios depending on the actual load.

The last plot of the first set shows the ratio between the detection ratio of OWL and that of LTEye: the two solutions decodes the same number of RB in 45% of the frames ($x = 1$), OWL never decodes less RBs than LTEye ($x = 0$), but consistently decodes a larger number of RBs in the majority of the frames, providing an improvement larger than a factor of 10 in 5% of the frames and an average improvement larger than a factor of 4.

Although mis-detection happens with higher probability, both solutions can also generate false positives, for instance due to strong noise/interference. However, since in our tests false positives have been detected in very few tests only, we deem their impact negligible.

Chapter 9

Fine-grained LTE Radio Link Estimation for Mobile Phones

Can we trust mobile phone data rate measurements? This apparently trivial question is key to evaluate the feasibility of the anticipatory networking [3] paradigm and the related future network solutions [182, 184]. For instance, exploiting achievable rate prediction to optimize mobile applications [7, 90, 106] requires some information exchange between mobiles and base stations so that current decisions (e.g. scheduling, admission control) can be made taking into account the future states of the system. However, while prediction errors have been studied [6, 63], the capability of mobile phones to obtain accurate measurements has never been investigated in mobile networks.

In addition to that, many recent studies [229–233] rely on crowd-sourced datasets to derive their conclusions without questioning mobile phone measurements accuracy and whether it is possible to aggregate them. Although reliable mobile phone applications to measure the network bandwidth exist [113, 234, 235], they focus on end-to-end measurements that do not provide the required level of granularity to enable anticipatory optimization. In fact, while end-to-end data rate is ideal to optimize TCP performance, the resource allocation optimization would rather benefit from the actual radio link data rate between eNodeB and UE.

In this chapter, we study whether mobile phones can accurately measure LTE radio link data rate and with which granularity (i.e. sampling frequency). To achieve this, we compare the data rate estimates computed at the physical layer of the radio link through a sniffer, at the mobile phone kernel through tcpdump and by a mobile application.

Our study is divided into two measurement campaigns: the first and largest set of experiments consists of burst transmissions, where a small amount of data is sent back-to-back to collect data rate estimates computed by the different entities (i.e., phone, sniffer and server), while in the second set, we evaluate latencies between single data packet transmissions and their corresponding Acknowledgment (ACK)s. These latencies allow us to study the root-causes of differences among the behaviors of different phones. In all the tests, we compared three mobile phones by different vendors and equipped with different chipsets, first performing the test from the server to the phone and, then, in the opposite direction. The main findings of our study are the following:

1. Mobile phones achieve accurate ($> 85\%$) and precise ($> 82\%$) data rate measurements with as few as 20 KB in the downlink, where accuracy and precision are related to how close the measurement are to the sniffer ground-truth readings.

2. Uplink measurements are less accurate and less precise (65% and 60% respectively in the worst case), because LTE uplink scheduling delay causes a higher variability in the results.

3. Different chipsets exhibit variable biases and performance, thus requiring dedicated calibration to optimize accuracy.

4. Downlink accuracy and precision are linked to the latency measured on the phone: chipsets providing shorter and more deterministic latencies obtain better estimates.

The rest of the chapter provides a survey of related work in Section 9.1, specifies the measurement setup and the devices involved in Section 9.2, and discusses the two measurement campaigns in Section 9.3 and 9.4. Sections 9.5 summarizes the main findings.

9.1. Related work

A considerable number of recent papers focus on LTE measurements and measurement techniques, but, to the best of our knowledge, none of them rely on accurate LTE scheduling information to validate their findings. Among them, Huang et al. [230] studied LTE performance measured from mobile phone data. In order to obtain a known reference for the results, the authors performed experiments using controlled traffic patterns to validate their findings.

The fraction of LTE resources used for communication is detected in [236] by means of power measurements. The goal of the authors is to evaluate the performance of M2M communications using experimental data. Similarly, RMon [219] is a solution to assess which resource blocks are used by comparing the average power measured over the resource bandwidth with that of the closest LTE reference signals. RMon achieves good performance and robustness, but it can only assess the average fraction of used resources. Hence, it cannot be used to capture the actual base station data rate.

LTEye [218] was the first attempt to decode the LTE control channel to access scheduling information. However the authors found in their later work [219] that LTEye could not provide sufficient reliability and a significant fraction of control messages remain undecoded. To overcome this limitation, we developed a reliable LTE control channel sniffer, called OWL [11]. In our tests, OWL successfully decoded 99.85% of LTE the control messages, thus obtaining a complete log of the eNodeB scheduling. MobileInsight [237] is a mobile phone application capable of accessing LTE control messages directly from the radio chipset and could also have been as an alternative to OWL.

A few papers [238–240] use commercial tools and/or operator network information to evaluate LTE performance, but their datasets (if released) only provide aggregate metrics that do not allow us to achieve the objective of this paper. The vast majority of papers however, just rely on measurement performed using mobile phones or replicated in laboratory experiments. Phone traces are used in [232] to evaluate network performance. The same authors developed a framework [233] to manage mobile phone measurements and a similar project was developed in [229]. In [231], LTE performance predictors are evaluated in laboratory setups. In addition, [241] uses TCPdump traces to perform energy efficiency evaluation of smartphones and [242] studies LTE shared access in a trial environment.

Finally, although not specifically developed for LTE, the following contributions discuss mobile measurements in general terms. The most popular approach is Ookla's Speedtest [234], which can provide a very accurate evaluation of the steady-state rate achievable by long-lived TCP connections. However, Speedtest is both data intensive (with fast connections, one test can consume more than a few tens of megabytes) and cannot provide estimates at the granularity required in this study. A few recent papers [113, 235] studied end-to-end achievable throughput, also accounting for inter-arrival times and passive monitoring techniques, but without comparing their findings with ground-truth readings. The accuracy of WiFi measurements performed by mobile phone is studied in [243] based on a timing analysis. However, their results cannot be applied to our scenario for two reasons: WiFi and LTE differs significantly in terms of scheduling and MAC protocols, and tcpdump traces do not provide a reliable ground truth for the physical radio link.

This study improves over the current state of the art by, first, evaluating data rate estimates on the radio link, instead of end-to-end throughput and, second, by relying on an accurate LTE sniffer to obtain a ground truth of the measurements.

9.2. Setup and Definitions

Figure 9.1 illustrates our experimental setup, which consists of five entities. The target UE is the mobile device under test which is connected to the target eNodeB. The sniffer is a BladeRF x40 software defined radio [215] that samples and records the LTE signal to be decoded by OWL. The sniffer is shown as connected to the eNodeB-UE link only, but it actually records and decodes all control messages sent by the eNodeB and, thus, it is aware of all of the traffic exchanged in the cell. The server is a PC in our local network configured with a public IP address in order to be reachable by the target UE. The Internet cloud in the figure groups all the links that form the backhaul of our setup including the operator network. Finally, the controller is a second PC in our local network which is directly connected to the target UE and the sniffer via USB and to the server via Ethernet.

In order to assess the impact of different hardware, we choose three mobile phones from different vendors with comparable technical specifications but equipped with different chipsets. In particular, we opt for a Motorola MotoG LTE [244], a Huawei P8 Lite [245] and a ZTE Blade

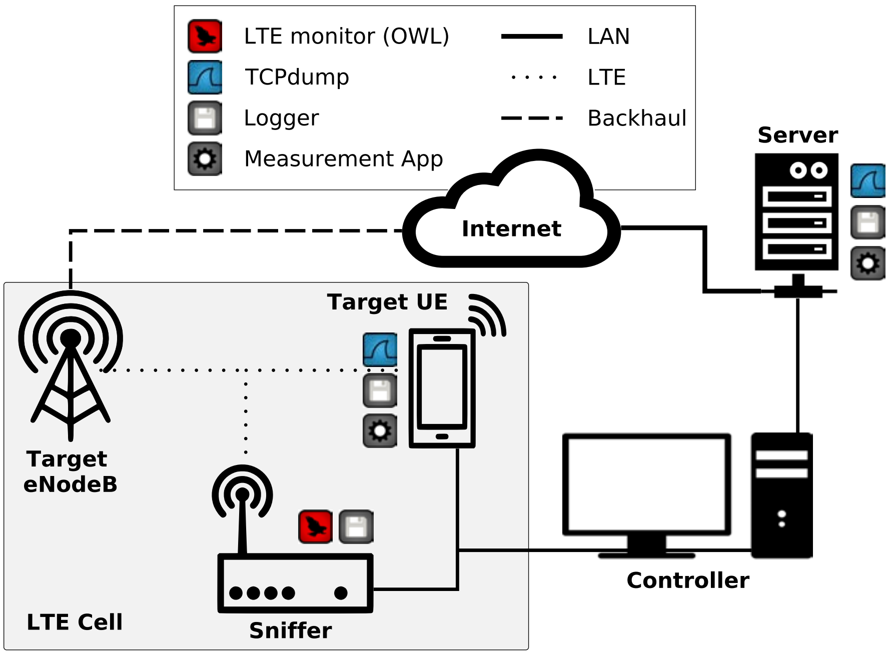


Figure 9.1: Experiment setup showing devices, connections and software.

A452 [246] equipped with Qualcomm, Huawei and MediaTek chipsets, respectively. The following list summarizes the features relevant for this study (the short names used in the rest of the paper are written in bold face):

- Motorola MotoG 4G (2014) – Chipset: Qualcomm Snapdragon 400 MSM8926; CPU: ARM Cor-tex-A7, 1200 MHz (4 cores); Android: 4.4.2 KitKat; RAM: 1 GB.
- Huawei P8 lite (2015) – Chipset: Huawei HiSilicon KIRIN 620; CPU: ARM Cortex-A53, 1200 MHz (8 cores); Android: 5.0.2 Lollipop; RAM: 2 GB.
- ZTE Blade A452 (2015) – Chipset: MediaTek MT6735P; CPU: ARM Cortex-A53, 1000 MHz (4 cores); Android: 5.1 Lollipop; RAM: 1GB.

We have four different software modules in our setup. The gear-shaped icon refers to the Measurement App, which controls the communication between the target UE and the server. For every successful socket call (either “send” or “receive”), it logs the time and the amount of data exchanged. This application is implemented in Python to obtain the same behavior both on the phone and the server. The shark-fin-shaped icon refers to TCPdump [247], which we use both on the UE and the server to obtain transmission timestamps at the kernel level as well as the payload size. The floppy-disk shaped icon illustrates the Logger application that formats the output of the other tools for later analysis.

The LTE monitor (owl-shaped icon) implements our Online Watcher for LTE (OWL [11]) control channel measurements. OWL is built starting from srs-LTE [214], an open-source implementation of LTE, and extends its functionalities to provide a reliable decoder of the physical control channel. From LTE control messages, OWL computes the transport block size assigned to each downlink and uplink communication. In this way, we can measure the actual LTE radio link data rate in every Transmission Time Interval (TTI), i.e. 1 ms. This data rate differs from the usual notion of end-to-end throughput and it is the main metric needed for anticipatory networking.

The LTE cell used during the tests belongs to Yoigo, a Spanish mobile network operator, and operates in LTE band 3 (1800 MHz) using a bandwidth of 10 MHz. The cell is chosen due of the relatively low load and the very good signal quality from the test location.

Our setup is characterized by three physical and five logical measurement points: we monitor the communications at the target UE, at the sniffer and at the server. Both the UE and the server collect information by means of TCPdump and at the application to capture the difference between application and kernel measurements by means of a data rate estimation technique using packet train dispersion [13].

As introduced above, we perform two measurement campaigns, the first dealing with burst transmission (see Section 9.3.1 and 9.3.2 for the test description and the results respectively) and the second with periodic isolated transmissions (Section 9.4.1 and 9.4.2). In the first campaign our goal is to evaluate the accuracy and the precision of fine-grained measurements, while in the second we study latencies in the different devices. Both campaigns consider both downlink (from the server to the UE) and uplink communication.

9.3. Burst Transmissions

The first measurement campaign has the main objectives of evaluating the accuracy and the precision of data rate estimates obtained by mobile applications, and to analyze the differences in performance obtained by the three phones. We use the following symbols: t, s, n and r denote durations, transmission sizes, number of packets, and the data rates. All these quantities are easy to compute from the information available in our tests and they do not require complex filtering. In fact, we just evaluate the data rate $r = s/t$ as the ratio between the amount of data s transmitted in a given time and the time t itself.

9.3.1. Experiment Description

We focus on packet trains (burst) from when they are first sent back-to-back from an application to their reception at the other endpoint. In particular, we are interested in comparing transmissions in the LTE radio link and the events tracked by a mobile phone at the application and the kernel level. We use Figure 9.2 as an example of a downlink test. The packets are generated by the application almost at the same time. As they are sent through a TCP socket they become spaced according to TCP dynamics and delays. For all layers, empty markers represent ACKs, except for the phone application layer where they mark packet receptions.

For the analysis, we define interarrival time t_I as the interval between two consecutive arrivals on the same layer and burst time t_B as the time between the first and the last packet of a train.

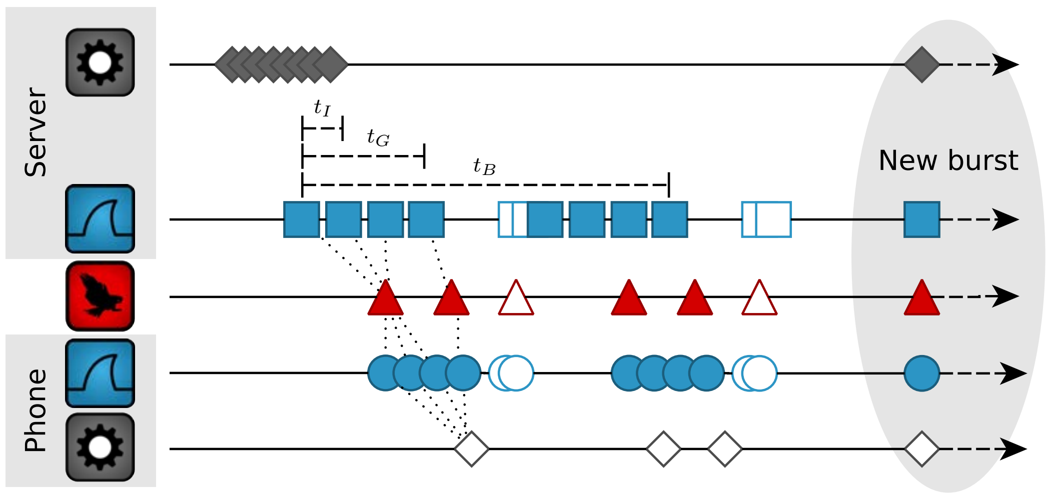


Figure 9.2: Communication diagram for downlink burst transmissions.

LTE may impose a further grouping of packets when large transport blocks can fit more than a single TCP packet; this is observed at the phone as a group of packets arriving almost at the same time and as a single event at the sniffer. We define group time t_G as the time elapsed between the first and the last packet of a series of continuous arrivals. The data rate computed on groups is the measure that approaches the most the physical rate. In what follows groups are identified by those packets whose interarrival times are shorter that a threshold $t_I \leq \tau \leq \tau_C$, where $\tau_C = 1$ ms is the TTI of LTE.

To compare LTE with phone and server traces, we fix the burst size to 100 and 30 KB for downlink and uplink experiments, respectively, to obtain at least 10 transmissions per burst: in our setup with a 10 MHz channel, the maximum LTE transport block size is 73392 and 28336 [227] bits in downlink and uplink, respectively.

9.3.2. Experiment Results

In this section we compare data rates measurements by means of an estimator ratio defined as $\eta = r/r_0$, where r_0 is the reference data rate, which, if not otherwise specified, is measured by the LTE sniffer. The estimators' accuracy is highest when the the ratio is $\eta = 1$ and degrades if it is either higher (overestimation) or lower (underestimation). Moreover, the standard deviation of the ratio is proportional to the estimator precision. Thus, we show the distribution of the estimators' ratios and we provide accuracy $\alpha = [1 - |1 - \overline{\eta}|]_0$ and precision $\rho = 1 - \sigma(\eta)$, where $\overline{x}$, $|x|$, $\sigma(x)$ are the empirical average, the absolute value and the standard deviation of x and $[x]_0$ is x if $x > 0$ and 0 otherwise. In the following results the overheads between the application and the kernel (about 3.95%) and between the kernel and the sniffer (about 0.8%) are compensated. The first and foremost results of our study are illustrated by Figure 9.3, which shows the empirical probability

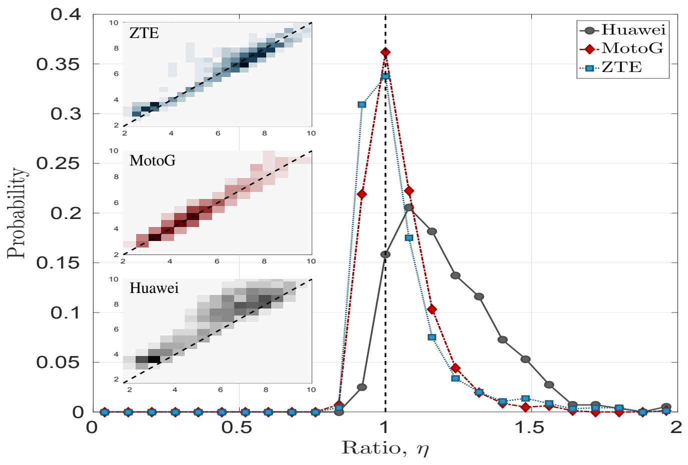
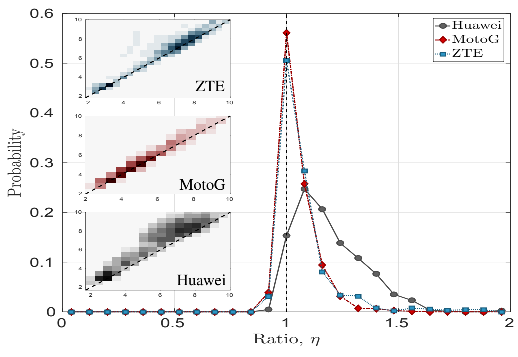


Figure 9.3: Comparison of the estimator ratios computed on burst by the application. The small plots on the left show estimator densities: the x-axis is the cell ground truth and the y-axis the estimate.

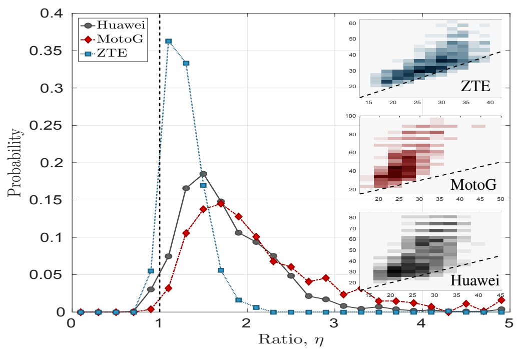
density function (epdf) of the estimator ratios obtained using burst by the three different phones computed by the application. The small plots on the left of the figures show the density of the estimators in a reference system where the x-axis reports the cell ground truth and the y-axis the estimate: the darker the color the more estimators are in that area. The black dashed lines in the plots show what an ideal estimator would achieve: the ratio distribution would be a single spike in 1 and the densities would be on the line $x = y$.

R1 – Phone applications can obtain accurate and precise downlink data rates measurements: Figure 9.3 demonstrates that the peaks of the epdfs are very close to 1. The three phones achieves accuracy of $\alpha = 85\%$ (Huawei), $\alpha = 96\%$ (MotoG), and $\alpha = 95\%$ (ZTE). The width of the edpfs is related to the estimators' precision, in particular, the three phones have $\rho = 89\%$ (MotoG), $\rho = 85\%$ (ZTE), and $\rho = 82\%$ (Huawei). The precision is also related to the width of the estimator clouds in the small plots – wider clouds corresponds to the lower precision.

R2 – Different phones have different biases: the slightly lower score of the Huawei phone means that it tends to overestimate the data rate by about a 10%, which can be easily compensated. The same results can be verified in the small plots: the MotoG's and ZTE's densities are centered on the $x = y$ line, while the Huawei's is slightly above. Since the estimators are obtained as size-over-time ratios and the bursts have fixed size, the root cause for accuracy and precision has to be looked for in the variability of the burst duration. In particular, if a phone consistently measures shorter burst times, it will overestimate the rate and, if the time measurements are variable (e.g., random delays due to different loads on the CPU) the corresponding precision will be lower. Thus, systematic errors impact the accuracy, while random errors affect the precision of the estimates. As a consequence, it is important to compensate for the biases of the different phones when dealing with crowd-sourced measurements, otherwise errors could accumulate unpredictably.



(a) Kernel–bursts



(b) Application–groups

Figure 9.4: Comparison of the estimator ratios computed on burst by the kernel (a) and on groups (b). The small plots on the left show estimator densities: the x-axis is the cell ground truth and the y-axis the estimate.

R3 – Accuracy and precision are independent of the actual data rates: examining the small plots in Figure 9.3 the estimators span the whole x and y axes between 2 and 10 Mbps. This means that, during the experiments, the network load varied so that the actual data rate achievable by our target phones (all of them show similar behavior) was changing. In addition, the actual data rate of the experiment does not affect the estimator quality. The slightly larger cloud at higher rates is expected since the same percentage error causes a larger absolute error at higher rates.

R4 – Kernel measurements are slightly more precise: Figure 9.4(a) shows the estimator ratios' epdfs when the measurements are performed by tcpdump. We expect these measurements to show better performance than those obtained from the application, since they are collected by tcpdump, they are time stamped when the kernel receives packets (through an interrupt from the chipset). However, not only are the precision improvements very small (i.e., 3%, 1% and 1% for MotoG, Huawei and ZTE respectively), but the accuracy scores are almost unchanged.

Figure 9.4(b) is obtained using groups instead of the longer burst. Since the transmissions within a group are expected to belong to consecutive radio link layer transmissions, the data rate estimate is likely to capture the exact rate used by the eNodeB. However, these measurements are more sensitive to timing precision. Since the group threshold for the cell is 1 ms, groups are characterized by transmissions in every TTI and, as soon as a single TTI is skipped, the group ends. Thus, if timing is not precise, two or more separate groups in the cell can be detected as a single group at the kernel or by the phone application. As a consequence, while it is possible to easily separate bursts and have a unique mapping between bursts in the different layers, this is not true for groups, whose composition is device and layer dependent.

R5 – Group-based data rate estimators are imprecise and biased: our measurements show that the accuracy (α) and the precision (ρ) of data rate estimators computed on groups are low. In particular, MotoG achieves $\alpha = 16\%$ and $\rho = 46\%$, Huawei $\alpha = 25\%$ and $\rho = 48\%$ and ZTE

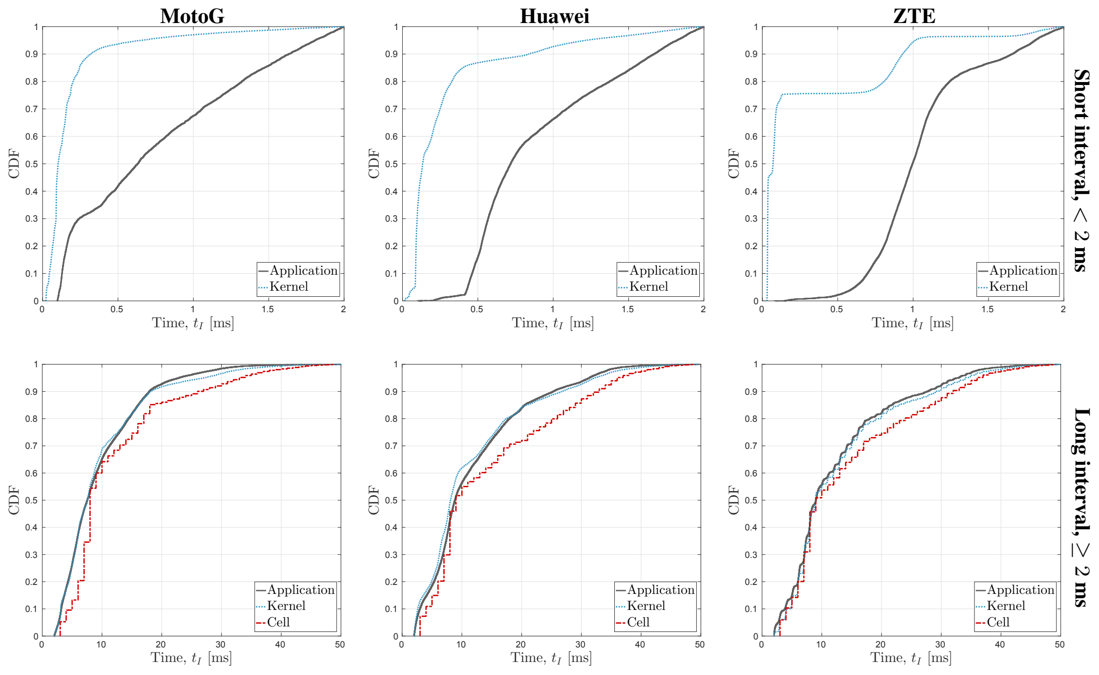


Figure 9.5: Interarrival time CDFs for short (top) and long (bottom) intervals and the three phones.

$\alpha = 75\%$ and $\rho = 79\%$. Instead, the ZTE phone measurements, even though overestimating by circa 25%, maintains a quite acceptable precision, close to 80%.

A close examination of the density plots reveal that group-based measurements have to estimate higher and more variable data rates, because they are not averaged over the longer duration of bursts. The ground truth data rate (x-axis) extends up to 45 Mbps, which is close to the maximum throughput in our setup. Moreover, all the phones overestimate the actual readings, reaching estimates even higher than the maximum reachable of our setup. This is a further proof of the importance of precise timing and deterministic latency in the phones to enable the most fine-grained estimation.

Before moving to uplink results, a few considerations on the packet interarrival times are in order. Figure 9.5 provides a set of graphs showing the Cumulative Distribution Function (CDF) of the interarrival times. To emphasizes the difference between the interarrival times of packets related to the same LTE transmission from those related to intervals separating continuous arrivals, we plot on the left column the CDFs for the interarrival times shorter than 2 ms (2 TTIs) and, those longer or equal to 2 ms and shorter than 50 ms on the right column. We don't show interarrival times longer than 50 ms, since those are almost always related to inter-burst rather than intra-burst arrivals. The matrix rows show from the top to the bottom, results for MotoG, Huawei and ZTE. We omitted the cell CDFs in the plots on the left, since it would have been a single spike at 1 ms.

Focusing on the left column, we can see that the CDFs of different phones and those obtained at the kernel and at the application are very different. For instance, the MotoG plot shows that the majority (90%) of interarrival times measured at the kernel are shorter than 0.3 ms, but only 30%

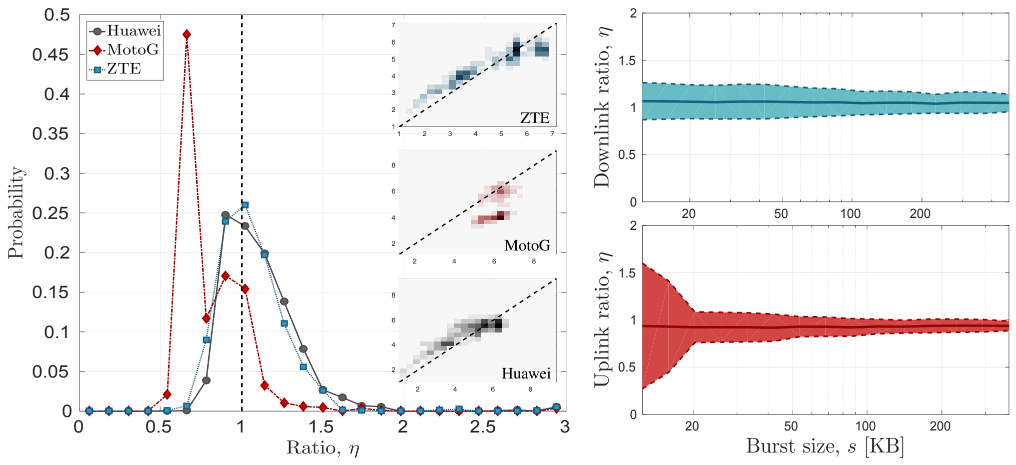


Figure 9.6: Estimator ratios computed on burst in the uplink (left). Estimator ratios against different burst size.

of those measured by the application are shorter than 0.3 ms. This means that multiple packets that are distinguishable at the kernel are received by the application as a single stream, thus, fixing the threshold $\tau = 0.3$ ms identify packets belonging to the same LTE transmission from those belonging to either the previous or the next.

Instead, the CDF of ZTE interarrival times at the kernel shows two flat regions, one before 0.15 ms and the second after 1 ms: this is caused by intra-transmission arrivals (the former) and inter-transmissions arrivals (the latter). Conversely, this noticeable distinction is not found in the application trace. Accordingly, we fix the grouping thresholds to different values: $\tau = 0.2$ ms for the kernel and $\tau = 1$ ms for the application.

Finally, the Huawei CDFs only show slight inflections at 0.4 ms (kernel) and 0.7 ms (application), but both are less marked than those of the other phones. We set the two thresholds accordingly. As will be more evident hereafter, a more skewed CDF with distinguishable intra- and inter-transmission thresholds corresponds to more deterministic latencies in the phone and, in turn, to better group data rate estimates. Analyzing the plots on the right, we can compare the interarrival time CDFs measured by the application, the kernel and the sniffer. Here we observe that the Huawei phone that has a slightly lower accuracy in terms of data rate estimation, and also shows a larger gap between the cell CDF and the other two, in particular between 10 and 30 ms. This confirms that data rate measurements are influenced by timing precision.

One final observation related to the ZTE phone is that both the application and the kernel CDFs show the same stair-shaped trend as in the cell CDF. Again, this is due to a lower variability of the ZTE latency, which will become more evident in the following second set of experiments.

Figure 9.6 (left) shows the uplink data rate estimator ratios of the three phones. Again, we compare kernel measurements on the phone against the sniffer's ground truth for the cell. Note that uplink application measurements would require a dedicated application that could intercept ACKs or especially designed to monitor the sending socket. The normal socket behavior is to accept send requests from the application until the transmission buffer is full and, since this buffer is usually larger than our data burst, the application can send to the socket a whole burst at once making it impossible to measure the data rate at the phone application.

R6 – Mobile phones can obtain accurate and precise uplink data rates measurements: although the MotoG underestimates the rate by about 30%, the other two phones have the peaks of their epdfs very close to 1. In particular, they achieve an accuracy α of 93% (Huawei) and 97% (ZTE), while MotoG stops at 65%. The edpfs are also wider than those related to the downlink. This is even more evident from the density plots on the right of the figure, which highlight that the precision of uplink measurements is lower than that obtained in the downlink: 73% for the Huawei and 74% for the ZTE. Not only does the MotoG have only 60% precision, but also its density plot shows two regions where the densities accumulate. This exhibits a binary behavior of the device that will become more evident in the next section where we analyze the phone latencies.

In addition, we compare the phone kernel to the server data rate measurements. Since the results obtained are very similar to those shown in Figure 9.6 (left) we omit the graphics. However, it is interesting that in our experiments, uplink burst can be measured both on the phone and the server achieving similar results. Since phone to cell measurements are taken before traversing the backhaul while phone to server results include it, we can conclude that the backhaul plays a minor role in our setup, because of the favorable location of the measurement server.

Figure 9.6 (right) reports the results obtained by varying the burst size from 10 KB to 1 MB for the downlink, and from 6 to 300 KB for the uplink. All the figures plot the average estimator ratio in the center of a shaded area that extends one standard deviation on each side. The figures are obtained by mixing together the results for all the phones.

R7 – Bursts of 20 KB provide high accuracy and high precision: the figures show that the estimator accuracy is independent of the burst size and the precision slowly improves with increasing size. Uplink proves to be more sensitive to very small burst (i.e., the shaded area is larger in the uplink plots for small bursts) as it is subject to more network randomness and it requires slightly longer transmissions. In contrast, in the downlink communication as few as two LTE transmissions are sufficient to obtain an accurate estimate. In our test we choose the minimum burst size to cause at least two transmissions at the maximum reachable data rate. In a larger bandwidth setup and when the next LTE releases will be deployed, the minimum burst size to achieve this results has to be increased proportionally to the maximum data rate.

R8 – UDP tests obtain the same results: the network provider used in our campaigns does not allow us to make reliable UDP tests, because of firewall and traffic shaping policies. To overcome this limitation, we repeated all the tests by emulating UDP by sending its packets with a TCP header through a raw socket. All the repetitions result in performance almost identical to that obtained by their TCP counterparts. The reason is that the measurement characteristics are dictated by the intra-burst timing, which, in turn, depend on the radio link technology, and not by the inter-burst timing, which, instead, depends on the protocol. Thus, radio link measurements only need for clearly separated burst for mobile phones to precisely estimate the data rate.

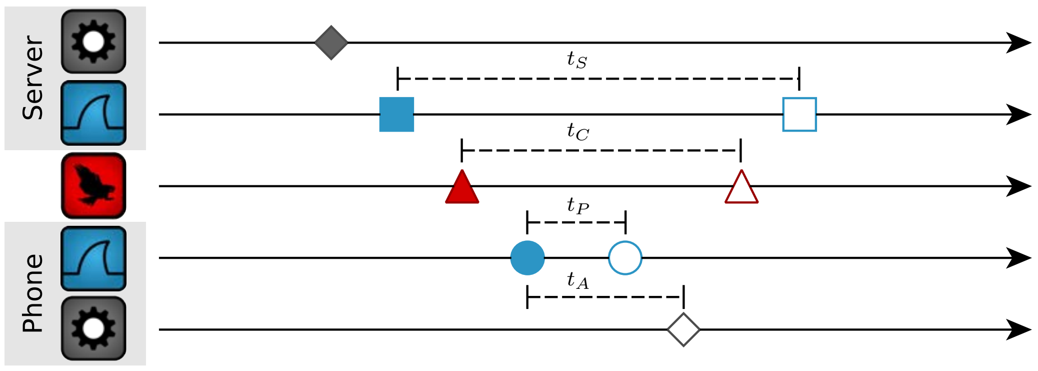


Figure 9.7: Communication diagram for the downlink isolated transmission. Dimension lines illustrate data-to-ack latency.

R9 – WiFi measurements are consistent, but different: we repeated the main tests on WiFi (IEEE 802.11g) by replacing OWL with a Warp Software Defined Radio [248] using the 802.11 reference design. We consistently observe that the performance obtained on WiFi are on the same order of magnitude of those obtained on LTE, but they are not identical in terms of bias and precision. Thus, while we agree on the main claims of [243] about overheads, we believe that different technologies require specific tests to evaluate their performance.

Before moving to the next set of experiments, we discuss a few more results for which we do not provide dedicated figures. We test our data rate measurements under three other conditions: 1) we stress the phone CPU to full load during the experiments; 2) we inject additional traffic in the cell under test up for to 95% load. Although, we expect the CPU load to add some delay to our measurement, we find that the phone kernel copes well with this load and we did not notice any significant change in the estimator performance. Similarly, the additional traffic injected in the cell only changed the actual measured data rates (i.e., lowering them), but did not decrease the estimator's accuracy.

9.4. Isolated Transmissions

This section details the second set of measurements. The objective of this campaign is to measure phone communication latencies to justify the differences in their behavior. As above, we first illustrate the experiment on a diagram (Figure 9.7) and on some trace examples and then we discuss the results.

9.4.1. Experiment Description

For the analysis of latencies measured at each communication layer we take particular care to link homologous events in the different measurement devices. While this is trivial in the phone and the server where we can access all packet header fields, identifying which LTE transmission contains a given packet in the scheduling log poses several problems. First of all, we need to find the correct RNTI of the target UE among the rest of the traffic, but while for burst transmission we could both rely on fixed burst size and periodicity, in isolated transmission tests a single packet is sent from the application, the payload of which should be large enough to differentiate it from LTE control messages and small enough to fit in a single transmit unit in the phone and the server interface. We fix the packet size to 500 bytes with a periodicity of 400 ms to leave enough time between subsequent repetitions not to confuse them with possible retransmissions. While other UEs scheduled together with our target may have similar periodicity, our UE could always be correctly detected.

Figure 9.7 shows the ideal communication diagram for the downlink isolated transmission test. Here, we monitor the time elapsed between each packet and the corresponding ACK. In what follows, we refer to this data-to-ack time as latency.

Although we only monitor relative times and we do not need a perfect synchronization between the measurement layers, in order to correctly couple events we make sure to capture the first event of each test in all the layers (to have a common reference) and, then, we run a causality check on each trace to compensate possible violations. Since each subsequent layer events have to occur after the events of the upper layer, we realign traces to follow causality.

Dimension lines illustrate latencies in the different layers with the only exception of t_A which refers to the time between a packet being captured by the kernel (phone for the downlink or server for the uplink) and when it is delivered to the application. Note that applications cannot measure their own latency without intercepting the communication ACKs at the kernel level.

9.4.2. Experiment Results

Figure 9.8 summarizes the main results of the latency measurements from which we draw conclusions about differences in the three phones' behaviors. All plots show the epdfs of the latency measured at the three measuring devices. All latencies are measured at the kernel level, since the application is not automatically notified of ACK receptions.

The plots in the figures are grouped vertically by communication direction and horizontally by layers and they are best read from the top left in clockwise order to follow the communication sequence.

The latency measured at the server in the downlink tests (top left) is the sum of the delays caused by two Internet traversals, two LTE scheduling delays (downlink first and then uplink) and phone processing (chipset time plus protocol stack traversal in the kernel). The latency at the cell downlink (top center) starts when the downlink LTE transmission is already scheduled and, as a consequence, it only contains the phone processing and the LTE uplink scheduling delays. The latency at the phone downlink (top right) starts from when the kernel receives the reception interrupt from the chipset to when the ACK transmission to the communication interface.

R10 – Chipsets with short and deterministic latency achieve more accurate and precise data rate estimation: in downlink tests, the latencies are similar in all the layers, except on the phone. The ZTE latencies exhibit a single peak before 0.5 ms, the Huawei a single, slightly wider

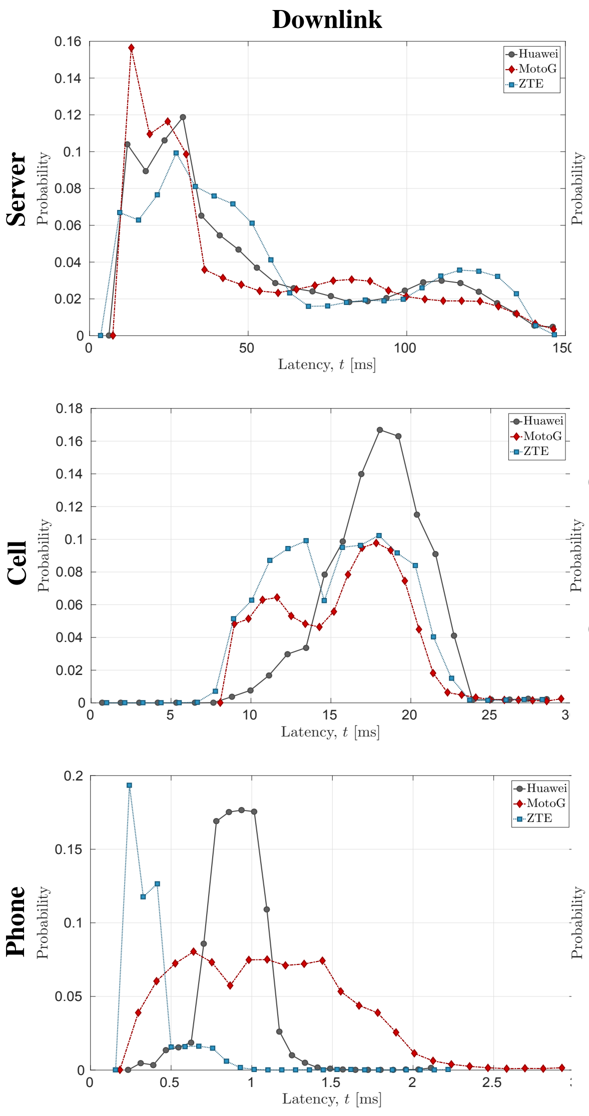


Figure 9.8: Empirical probability density functions of the latencies.

peak at about 0.9 ms, while the MotoG shows a wider distribution of its latencies ranging from 0.3 to almost 2 ms. Recalling from Figure 9.3 that MotoG and ZTE achieve higher accuracy and precision in their data rate estimates, we can conclude that chipsets with a shorter latency are more accurate in estimating the data rate. Instead, the Huawei latency is closer to the length of the LTE TTI and is the cause of the overestimation of the data rate. To explain the difference in performance between the ZTE and MotoG, we need to consider the CDFs of their short interarrival time (recall Figure 9.5 top and bottom graphs on the left). While the MotoG application captures intra-group events (shorter than 0.5 ms), the ZTE application distribution starts only after 0.5 ms, but it is very precise at the kernel. Thus, the MotoG application data rate estimate fares better than ZTE, which, instead, is more precise at the kernel level only.

The low variability of ZTE latencies explains why the ZTE long interarrival time distribution has the same stair-shaped trend as the cell distribution. As a consequence of this higher precision at the kernel, the ZTE phone can better discriminate between LTE transmissions, which, instead, are smoothed in the other two phones, and is able to obtain more accurate group-based data rate estimates (recall Figure 9.4(b)). Also, since the three phones show similar times for server and cell latencies we can exclude that network traversals and LTE scheduling impact data rate estimates between phone and cell in our setup.

R11 – LTE discontinuous reception configuration [249] influences uplink data rate estimates: the bottom row of Figure 9.8 illustrates the epdfs of uplink experiments. Latencies measured on the phone kernel include both uplink and downlink LTE scheduling, two Internet traversals and the server processing; latencies at the cell include the Internet traversals and the server processing delays, while the server latencies only include processing delay. The server processing is negligible, since it is shorter than $50\,\mu s$. Similarly, we can exclude that the network delays play an important role in the uplink data rate estimates, since the three phones show almost identical latencies when measured at the cell. Conversely it is the LTE uplink scheduling delay that influences the estimator the most. This delay is expected to be about 20 ms for a connected device starting a new transmission, while in our measurements the epdfs are centered at about 50 ms (Huawei and ZTE) and 85 ms (MotoG) with a smaller peak at 40 ms. All these longer uplink delays are due to LTE discontinuous reception (DRX), which is an energy saving feature that allows mobile phones to duty-cycle between sleep and wake phases. Since to discriminate among different transmissions we separate them by 400 ms (1 s for bursts), all the transmissions start with the devices in DRX mode. The actual duration of the sleep time depends on agreements between UE capabilities and eNodeB requirements. Thus, MotoG uses a more conservative DRX setup (with a longer sleeping period), most likely due to the fact that it is an earlier (2014) model than the other phones (2015). The overall effect of this latency is that uplink data rate estimators are less precise than downlink estimates by circa 10% due to the wider distributions of the latencies.

9.5. Summary

Figure 9.9 provides a visual summary of the results discussed in the paper. In the figure, one boxplot is shown for each of the main experiments, highlighting the median (central mark of the boxes) and the 25th and 75th percentiles (box edges) of the estimator ratios η. At the bottom of the figure we specify the type of η used. BA are ratios between data rate measurements performed on bursts (B) at the application (A) and cell estimate. BP are the same but computed by the phone kernel (P), while GA are the same as BA, but computed on groups (G). All downlink ratios use the sniffer as a reference. On the uplink side, we show PC, which compares phone kernel (P) estimates with cell (C) and PS which use the server (S) application as a reference. All uplink results are computed on bursts. Note that we do not show graphs for PS in the previous results, since PC and PS are quite similar. This shows that in our setup uplink data rate estimates on the server and the phone obtain comparable results.

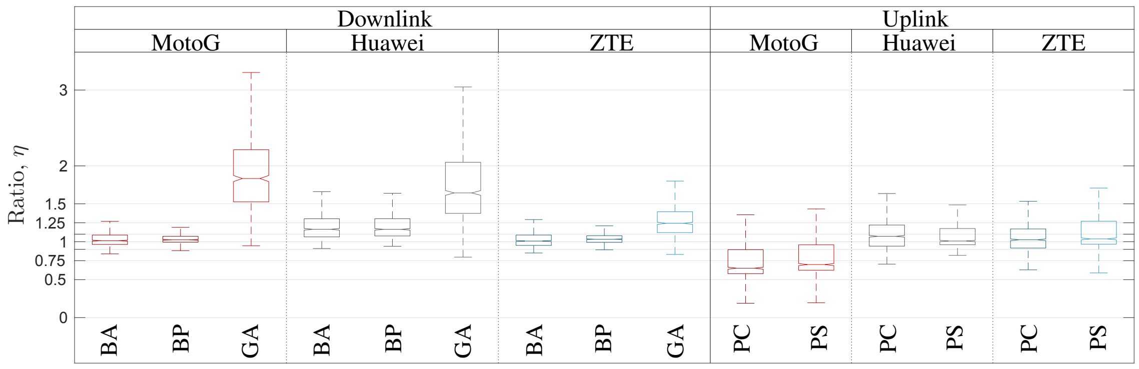


Figure 9.9: Overall comparison among the data rate estimators. All boxplots show the distribution of the estimator ratios from the 25^{th} and 75^{th} percentiles.

Looking at all the results side-by-side, it is evident that on the one hand side, all the phones are capable of accurate and precise data rate estimation, but on the other hand they have significantly different biases and precisions, as seen for instance, with Huawei and ZTE for BA in the downlink and for PC in the uplink. Similarly noticeable is that group-based estimators, GA, achieve reasonable accuracy on the ZTE phone only and the MotoG uplink measurements are heavily impaired by the different LTE latency.

To conclude our study on LTE radio link estimation, we can affirm that the precision and the accuracy achieved by the three devices are sufficiently high to enable anticipatory networking optimization up to a time granularity of about 50-100 ms and after having compensated the device bias. This is true for both uplink and downlink estimates either obtained by the kernel or the application. Conversely, to increase the measurement granularity and, in turn, the optimization potential, direct readings of the actual physical rate are needed.

Part IV : Practical Evaluation and Conclusions

Summary

This fourth and last part of this thesis presents our practical validation of anticipatory networking solutions in Chapter 10 and the global conclusions of the thesis in Chapter 11.

Chapter 10 describes the measurement campaign we performed in four locations in Madrid and Leganes using OWL (see Chapter 8). In particular, we analyze the data both as cell-based aggregated information and on a per-user basis, identifying characteristics that can support or hinder prediction and optimization. In this chapter, we also present a comprehensive framework that encompasses the different prediction-based techniques discussed so far and combine them with different level of prediction accuracy. This framework is then used to evaluate anticipatory networking performance on real data.

The work presented in this chapter is, to the best of our knowledge, the first complete evaluation of anticipatory networking on real data. In fact, while many previous works showed the theoretical benefits of assisting the optimization process with system state prediction, we evaluate how this approaches perform in practice. In addition, we consider several variants of them showing the effects of different design choices on system performance and user experience. We compare omniscient predictors and realistic models that can be implemented in base stations or mobile phones, operator-driven and user-driven optimization, and resource preserving and data rate maximizing approaches.

This final practical validation shows that not only is anticipatory networking an effective theoretical approach, but that it also is implementable in practice in both current networks, as our LTE tests show, and in future generation networks, where even better improvements are expected due to the always increasing computational capabilities of portable devices and communication systems in general. Chapter 11 concludes the thesis summarizing its contributions and the obtained results.

Chapter 10

Data-driven Evaluation of Prediction-based Optimization methods

The main missing element in the whole body of work about anticipatory networking is an in-depth evaluation of how predictive optimization would perform in the real world. In this chapter, we fill this gap by applying prediction-based optimization to the resource allocation data of LTE networks that we collected in four locations in Madrid over one month. In particular, we identify in the whole dataset those traces (i.e., almost continuous data flows belonging to a single user) that are suitable to be predicted. For these traces, we allow the data transfers to be re-organized so that future exchanges can be anticipated (i.e., buffered) if that improves a given objective function.

In particular, in our evaluation we treat all traffic that exhibits good predictability as elastic (i.e., it can be buffered in advance) and the rest as background traffic, which translates to a fixed and unpredictable load for the cell. This assumption allows us to study how the network would have performed, had it prediction capabilities. While not all predictable traffic is elastic, this is true for much of the high volume traffic such as video. Another important research direction (beyond the scope of this paper) is to make application traffic more elastic and provide means to signal delay requirements to the network.

Our analysis shows that omniscient optimizers can improve the average network efficiency by 35-40% in both communication directions, and more than doubles the data rate for downlink communication only (uplink data rate can be increase by circa 8% only, because of a smaller margin of improvements). The performance obtained using realistic predictors show that anticipatory solutions are both feasible and effective, even though the performance are between 5 and 10% worse than the optimal. This confirms the preliminary results obtained in the literature over synthetic datasets and the benefit that predictive optimization can bring to next generation mobile networks.

In the rest of the chapter, after a review of the related work in Section 10.1, we discuss the following novel contributions. Section 10.2 illustrates the comprehensive anticipatory networking framework we used to evaluate the datasets. The section provides details about 1) time series prediction, 2) linear programming formulations to minimize network resources and maximize users' data rate, and 3) the complete optimization framework that encompasses prediction accuracy and objective functions. It also explains how to proceed from data collection to performance evaluation. Section 10.3 discusses our measurement campaign providing 1) a summary of the LTE characteristics, 2) a short description of the datasets, and 3) a preliminary analysis on the dataset where we distinguish the predictable (and thus optimizable) components from background traffic. Section 10.4 examines the results obtained by the different anticipatory networking techniques on the datasets and provides further considerations about them and anticipatory networking in general.

10.1. Related work

In this section we discuss a few alternative approaches to our evaluation framework, alternative tools to record mobile network traffic and measurement-driven analysis of mobile networks. Yin et al. [90, 106] propose a throughput prediction solution based on clustering and hidden Markov models. Their predictor is subsequently used to control video bitrate selection in a multimedia streaming application. Finally, they evaluate their approach on a proprietary large dataset provided by a Chinese commercial video provider. Similarly, Kurdoglu et al. [69] exploits an online linear adaptive filter to optimize the video bitrate by controlling the bit budget thanks to future capacity prediction. Muppirisetty et al. [76] investigate the spatial prediction of wireless channels using Gaussian processes. Atawia et al. [65] focus on energy savings obtained thanks to predictive resource allocation and uncertainty management. Also, Yu et al. [94] optimize energy consumption by means of predictive scheduling of multi-technology wireless networks (i.e., WiFi and cellular), which is based on Lyapunov optimization. Finally, Du at al. [97] design a predictive backpressure algorithm to solve the resource allocation problem for multimedia streaming.

These are just a few of the many papers adopting anticipatory networking and we encourage the interested reader to read further on the topic [3], where we provide a thorough review of the state of the art. The framework described here is not meant to provide yet another variation on the topic, but allowed us to test the performance of many realistic approaches against theoretical bounds on a big dataset.

Most of the literature on anticipatory networking, but a few exceptions (e.g., [90]) evaluate their solutions on synthetic, even though realistic, datasets. This is mainly due to the unwillingness of mobile operator to share their traffic information with research centers and universities and to the fact that public datasets are limited in terms of traces length and data size, thus making them hardly usable for our objectives.

If mobile operators disclose their datasets, very interesting and insightful papers originate. For instance, the recent works of Furno et al. [250, 251] study the influence of human activities on mobile communications and identify several traffic patterns that can be used to enhance anticipatory networking. In a similar fashion, Wang et al. [252] analyze the traffic in Shangai and conclude that there are five main traffic profiles that represent most of the activity in the 9000+ studied cells. The same dataset is also analyzed by Ding et al [253] to model the network capability. Previous studies, such as those of Shafiq et al. [112, 254] and Keralapura et al. [255] investigate traffic profiles and their predictability.

Differently from all these studies, we built our dataset using our LTE sniffer [11] and we plan to make our dataset available to the community to allow for comparative studies and the development of practical solutions. Our dataset, which is intrinsically anonymous due to the use of temporary identifiers instead of unique user IDs, is also the only one to provide scheduling information at millisecond granularity. Thus, to the best of our knowledge, our dataset is the only archive of mobile network traffic obtained independently of mobile operators.

10.2. Anticipatory Optimization Framework

Anticipatory networking solutions include two main components: prediction and optimization. Here, we limit ourself to a few selected methods that allow us to evaluate both the maximum achievable gains due to anticipatory networking and the improvements that realistic solutions would achieve in the real world. We acknowledge that, depending on contextual information used and the application objectives, other solutions can exist achieving different performance. However, our methodology proved to be adequate to solve our optimization problems in very large datasets and shed some light on the actual performance of anticipatory networking solutions. For a more detailed review of possible applications and variants of these components we refer the reader to [3].

10.2.1. Optimization Problem

We use [8] as a basis for our optimization problem, which is defined as a centralized decision making problem, where a set $\mathcal{N}$ of N users share a given quantity of network resources over a set $\mathcal{T}$ of T time slots, also referred to as optimization window. The objective of our formulation is to assign the available network resources so that all users obtain the requested information while the cost for the network is minimized. We use the following inputs for the problem:

- Predicted achievable rate $r_{i,j} \in [0, r_M]$ is the prediction of the rate a user would achieve if no other user is scheduled. r_M is the maximum achievable data rate.

- Requirement $d_{i,j} \in [0, q_M]$ is the minimum amount of bytes needed in a given slot to stream the content at the minimum bitrate with no interruptions.

The problem is characterized by the following variables:

- Resource assignment $a_{i,j} \in [0, 1]$ represents the average fraction of resources assigned to user i in slot j. In each slot, each user can be assigned at most the total available rate, $0 \leq a_{i,j} \leq 1$, and the sum cannot exceed the total available resources, $0 \leq \sum_{i \in \mathcal{N}} a_{i,j} \leq 1$.

- Buffer state $b_{i,j} \in [0, b_M]$ tracks the amount of bytes stored in the buffer and b_M is the buffer size in bytes.

- Outage $l_{i,j} \in [0, q_M]$ is the missing data to fulfill the minimum content requirement $d_{i,j}$:

$$l_{i,j} = [d_{i,j} - b_{i,j} - a_{i,j} r_{i,j}]_0^{d_{i,j}} \tag{10.1}$$

where $[x]_a^b = \min\{\max\{x, a\}, b\}$ is a bounding operator that forces the undelivered quantity to be greater than zero and smaller than the requirement in the slot.

In each slot j user i receives $a_{i,j} r_{i,j}$, which can be used either to satisfy the requirements in the current slot or to fill the buffer for later use. Thus we can write the following equation that describes the next buffer state:

$$b_{i,j+1} = b_{i,j} + a_{i,j} r_{i,j} - d_{i,j} + l_{i,j}. \tag{10.2}$$

We define $b_{i,0}$ as the initial status of the buffer of user i.

In addition, we introduce three KPIs that we will use to build the objective function for our problem. Namely, we define the amount of used resources $\delta_i = \frac{1}{T} \sum_{k \in \mathcal{T}} a_{i,k}$, the fraction of continuous streaming time $\lambda_i = \frac{1}{T} \sum_{k \in \mathcal{T}} \left(1 - l_{i,k} d'_{i,k}\right)$ and the fraction of the extra data rate obtained $\theta_i = \frac{1}{T} \sum_{k \in \mathcal{T}} \left(a_{i,k} r_{i,j} d'_{i,j} - 1\right)$, where we use $d'_{i,j} = 1/d_{i,j}$ if $d_{i,j} > 0$ and 0 otherwise to avoid division by zero.

Finally, we build two objective functions: the first minimizes the network resources spent, while the second maximizes the overall delivered data. Both objective functions must guarantee minimum outage before tackling the specific objective: if resources are not sufficient to satisfy the minimum requirements, both functions will give the same resulting allocation, which minimizes the overall outage. For the **resource minimization** we obtain the following LP formulation:

$$\begin{aligned} \underset{A,B,L}{\text{minimize}} & \sum_{k \in \mathcal{N}} (\delta_k - K \lambda_k) \\ \text{subject to:} & a_{i,j} \geq 0; \quad \sum_{k \in \mathcal{N}} a_{k,j} \leq 1 - a_{B,j} \\ & l_{i,j} \geq 0; \quad b_{i,j} \leq b_M \\ & l_{i,j} \geq d_{i,j} - a_{i,j} r_{i,j} - b_{i,j} \\ & \forall i \in \mathcal{N}; j \in \mathcal{T} \end{aligned} \tag{10.3}$$

where the weight K ensures that the solver's priority is on outage minimization and $a_{B,j}$ represents the fraction of resources used by background traffic at time j. We refer to background traffic to those resources used for real-time or inelastic traffic, which cannot be moved and, thus, cannot be optimized. The **data rate maximization** LP is given by:

$$\begin{aligned}
&\underset{A,B,L}{\text{maximize}} \sum_{k\in\mathcal{N}} (\theta_k + K\lambda_k) \qquad (10.4)\\
&\text{subject to:} a_{i,j} \geq 0; \quad \sum_{k\in\mathcal{N}} a_{k,j} \leq 1 - a_{B,j}\\
&\qquad \sum_{k\in\mathcal{T}} a_{i,k} \leq a_{i,0}; \quad l_{i,j} \geq 0; \quad b_{i,j} \leq b_M\\
&\qquad l_{i,j} \geq d_{i,j} - a_{i,j} r_{i,j} - b_{i,j}\\
&\qquad \forall i \in \mathcal{N}; j \in \mathcal{T}
\end{aligned}$$

where $a_{i,0}$ is an upper limit to the total resources assigned to user i. Formally, the two optimization problems should have used mixed-integer formulations, because LTE resources are only assignable in finite quantities. However, since the time slots used for our optimization are two orders of magnitude longer than LTE TTI, the expected approximation error is smaller than 1%.

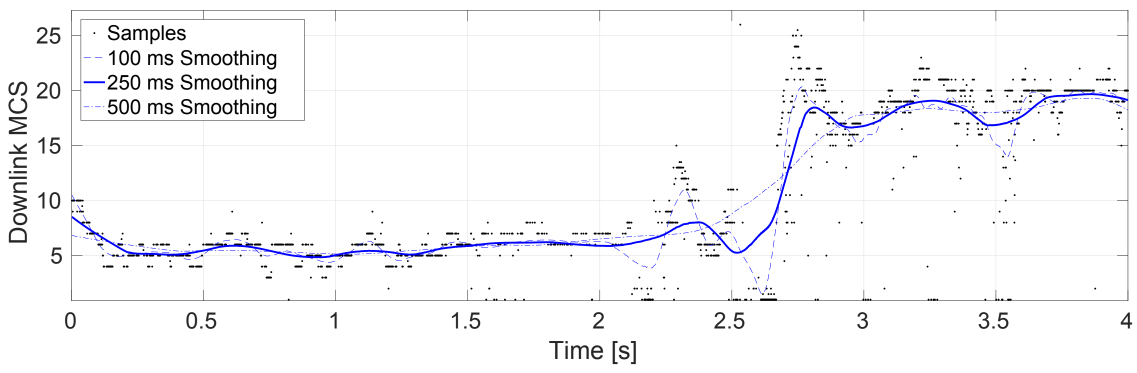
10.2.2. Prediction Methodology

Among the many prediction techniques, we opt for time-series analysis, because it is simple to implement, to train and their computational complexity is sufficiently low. Here, we make no attempt to compare different prediction schemes and we do not claim the superiority of the methods used here, compared to other solutions. Our objective is to show a feasible solution that can be easily adopted in current networks. In addition, we evaluate the impact of prediction errors over the optimization quality.

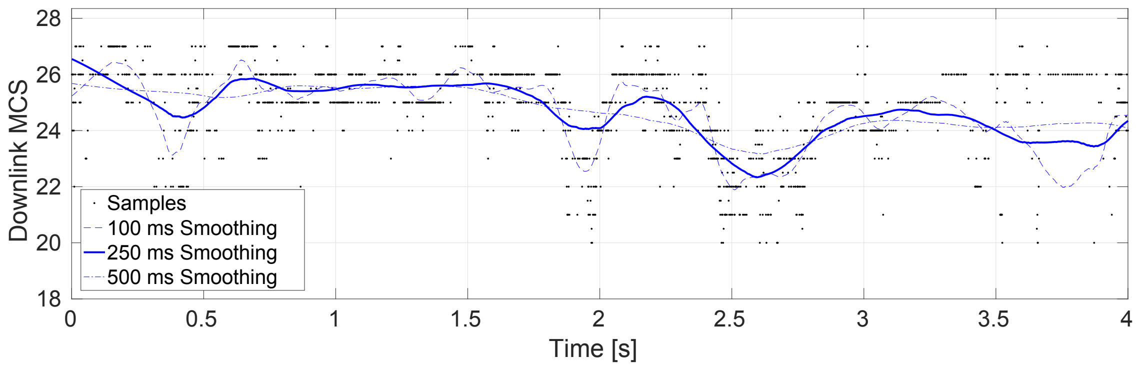
According to previous optimization solutions [6–8], we need to predict users' achievable data rate, because by knowing the maximum data rate all users can be assigned at any given time allows to optimize the resource allocation process. Achievable rate is a function of the MCS obtained using standard LTE tables [220]. Thus, in our measurement campaign we need to collect and study MCS traces together with resources assigned to all the users and their achieved data rate.

We adopt ARIMA time-series analysis to model each of the traces and, subsequently, we use the obtained models to evaluate the prediction Mean Square Error (MSE). Since ARIMA models requires the time-series to have equidistant samples in time, before applying the model we regularize our traces: first, we analyze the average MCS over time bins and, then, we linearly interpolate our traces over gaps longer than one bin duration (i.e., when a given trace contains no information over a period longer than a bin). We fix the bin duration to 200 ms which allows reliable achievable rate estimation [256] while preserving the MCS variability induced by user movements. In addition, the selected bin duration should be long enough to filter fast MCS variation due to fast fading in most scenarios.

To verify the impact of linear interpolation over unknown gaps we test it over very dense traces collected with MobileInsight [237] and we create gaps to be filled by linear interpolation.



(a) Vehicular



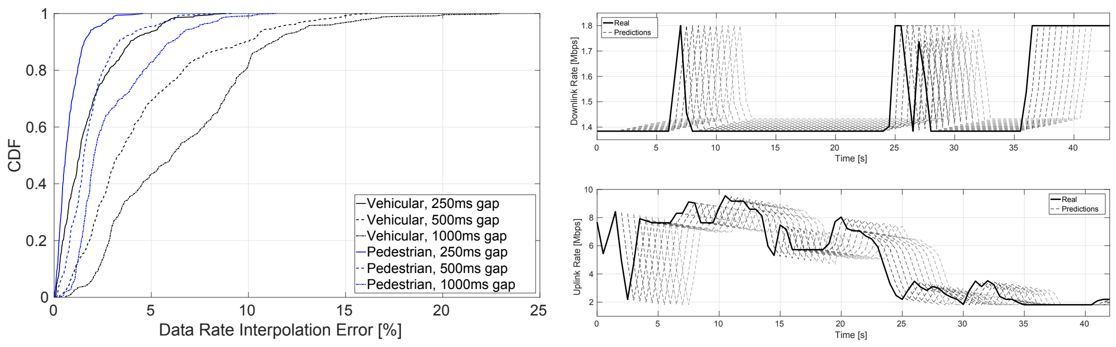
(b) Pedestrian

Figure 10.1: Two examples of traces captured with MobileInsight showing different level of smoothing.

The test traces are recorded either while walking at a regular pace or during car rides in the city center. In both scenarios the tested mobile phone is constantly receiving a video in order to ensure a dense trace (i.e., very frequent communications). Figure 10.1 shows two examples of the vehicular and pedestrian mobility effect on the MCS variation. Both figures shows instantaneous samples (black dots) and the effect of smoothing of different sizes (blue lines). Our tests, which are summarized in Figure 10.2(a), show that the error caused by linear interpolation is usually smaller than 5%, increasing substantially (max. 15%) only for long gaps and vehicular mobility.

We acknowledge that during information gaps anything can happen, but linear interpolation is the most viable no-nonsense approach to analyze our dataset without resorting to very complex mathematical models that would require an even longer computation time. In addition, we expect the users of the selected locations to exhibit either a pedestrian or a slow vehicular mobility, both of which can be approximate with linear mobility over time intervals as short as the one found in our traces.

An ARIMA model is characterized by three parameters: the autoregressive order p, the moving average order q and the degree of differencing d. For each of the traces, we choose the best orders for the ARIMA model according to the Box-Jenkins [200] methodology. Then, we estimate the model coefficients by means of least square regression. Note that we create a model for



(a) Interpolation error (b) Prediction Examples

Figure 10.2: Fig.(a): Cumulative Density Function of the error introduced by interpolating over gaps of varying size. Fig.(b): Prediction examples computed over downlink (top) and uplink (bottom) traces of the same user using an ARIMA model. The trace and the predictions are sampled every 0.5 seconds for readability.

each of the traces using all the information available for that trace. This allows to evaluate the best possible prediction obtainable with this methodology. In a real system, it might be impossible to have separate predictors per individual users and general models associated to user profiles might be used instead. Figure 10.2(b) shows an example of prediction obtained with an ARIMA model over a 40-second trace on the downlink (top) and uplink (bottom) channels. In both examples the model is able to extract a general trend (i.e., increasing for the downlink and decreasing for the uplink), but it is less effective during fast variations (e.g., second 25 in downlink).

10.2.3. Evaluation Framework

In the previous parts of this section we defined our prediction and optimization tools. We remark that the reasons for our choices were mainly twofold: 1) test optimality (with perfect prediction and LP optimization) against suboptimal and more realistic options and 2) control the computational complexity to evaluate them on our dataset. In particular, we define the following features.

We include three levels of prediction accuracy:

- **Perfect:** the exact achievable rates are fed to the optimizer.

- **Proactive:** the prediction is computed by feeding the ARIMA models defined above with all the past samples of the trace. Since the optimizer can accurately know a given user achievable rate only when that user is actively using the medium, this type of prediction requires some sort of active achievable rate measurements when the user is not scheduled.

- **Reactive:** the prediction is still computed using the same ARIMA models, however, past information is only updated when the user is scheduled. To feed the optimizer with a continuous trace we fill the gaps by linear interpolation and we feedback the predictor output as it past input until a new scheduling event happens.

Note that, both the proactive and the reactive prediction types require to recompute both prediction and optimization at each time slot, in order to account for updated information.

We analyze two objective functions:

- **Resource Minimization:** we use the problem definition of Eq. 10.3 to compute the minimum amount of resources needed to provide each active user in the system with the same total rate they obtained in the original dataset. We enforce causality, by allowing users to use resources in the past to satisfy requirements in the future, but not vice versa.

- **Rate Maximization:** we use the problem definition of Eq. 10.4 to compute the maximum data rate that could be obtained by each active user in the system exploiting the same total quantity of resources. The parameter $a_{i,0}$ is set to match the original resource quantity consumed before the optimization.

we consider two optimization types:

- **Centralized:** the two problem formulations above are already defined as centralized problem were a common solver uses all the available information to compute the best resource allocation.

- **Distributed:** in a distributed scheme each users optimize her behavior (i.e., the amount of requested data) according to her limited view of the system. The problem formulation is the same as per the centralized type, but it is repeated for each user individually.

The main difference between these two types is that centralized optimization cannot generate infeasible conditions, while the distributed type might create allocations overflows: this condition happens when a number of users decide to request data in the same time slot and their combined request is larger than the available resources. This is avoided in the centralized approach, by bounding the total request at any given time, but cannot be avoided in the distributed version, because users are not aware of other users decisions. When an infeasible allocation is decided, we normalize the requests proportionally and we adjust all the remaining parameters accordingly. However, by doing so, the users receive less than what requested and this may cause an interruption in the service being offered.

To illustrate the impact of prediction uncertainty, Figure 10.3 shows a detailed comparison example between a solution obtained by the ideal optimizer (solid black line) and one using the reactive prediction (dashed blue line). Both lines show the total data transferred, while the shaded area represents the downlink achievable rate. While the ideal solution only transfer data when the achievable rate is maximum, the solution adopting a reactive predictor cannot take full advantage of the best conditions because they happens too suddenly to be predicted accurately.

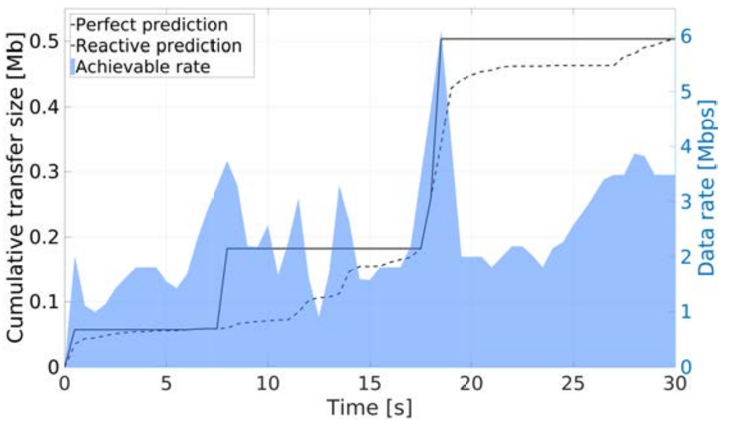


Figure 10.3: A comparison between the solutions obtained with perfect and realistic predictors.

We remark that transmitting when the achievable rate is higher means using the higher MCS and, thus, transmitting more with the same network resources.

In order to apply our evaluation framework on real data we proceed as follows:

1. Collect LTE scheduling information: we describe the tools we use and the locations where we perform the measurements.
2. Identify the predictable fraction of the traffic: active users exhibits characteristic features that help us distinguishing their trace from background/passive traffic.
3. Apply our evaluation framework on the obtained datasets.

10.3. LTE Measurements

We performed a one month measurement campaign in four LTE cells in Madrid. To collect the data, we used our Online Watcher for LTE (OWL) [11], a decoder of the LTE control channel. OWL uses a software-defined radio (SDR) to sample the LTE downlink channel and implements the decoding functionalities based on srsLTE [214], an open-source LTE library.

LTE scheduling measurements are possible because of centralized communication management and unencrypted control channel information. Centralized communications imply that a single base station, also known as eNodeB, coordinates the data transfers of the mobile phones, also known as user equipments (UEs), in both downlink and uplink channels. In particular, the eNodeB sends scheduling information to UEs using a dedicated channel. Thanks to our sniffer we are able to decode from the control channel the following information: 1) temporary user ID (C-RNTI) that does not allow to uniquely identify the user, but is sufficient to follow the scheduling of a given user over time until she stops her communications for longer than 10 seconds or she changes the cell, 2) assigned MCS, 3) allocated number of resource blocks, 4) transport block size. See Chapter 8 for further details.

Table 10.1: Dataset Statistics

	Callao	Rastro	Leganes	IMDEA
Operator	Movistar	Vodafone	Yoigo	Vodafone
Bandwidth	15 MHz	10 MHz	10 MHz	10 MHz
Frequency	1.8 GHz	800 MHz	1.8 GHz	800 MHz
Compressed Size	60 GB	19 GB	4 GB	24 GB
Total Time	35.5 days	37.5 days	21.3 days	18.7 days
Total Download	4.5 PB	0.86 PB	1.1 PB	0.15 PB
Total Upload	1.5 PB	0.3 PB	0.43 PB	0.02 PB
Total Traces	10.8 M	1 M	1.45 M	0.16 M
Active Traces	3.7 M	0.4 M	0.52 M	0.08 M
Median Downlink Load	5 %	1 %	2.5 %	0.1 %
Median Downlink Rate	1.13 Mbps	0.04 Mbps	0.24 Mbps	0.01 Mbps
Max Downlink Rate	21.3 Mbps	19.5 Mbps	22.2 Mbps	6 Mbps
Active Median Downlink Rate	12.2 Mbps	12 Mbps	9.6 Mbps	14.1 Mbps
Active Max Downlink Rate	110 Mbps	75 Mbps	75 Mbps	75 Mbps
Median Uplink Load	2.5 %	1 %	3 %	0.05 %
Median Uplink Rate	0.36 Mbps	0.06 Mbps	0.16 Mbps	5 Kbps
Max Uplink Rate	18 Mbps	12 Mbps	12.3 Mbps	4.9 Mbps
Active Median Uplink Rate	4.8 Mbps	2.7 Mbps	2.7 Mbps	2.3 Mbps
Active Max Uplink Rate	55 Mbps	37 Mbps	37 Mbps	37 Mbps

10.3.1. Campaign description

Our measurement campaign consists of the data collected by OWL during one month in four different locations. We selected the four locations in order to analyze how optimization methods would performs in areas with different uses (e.g. residential, commercial, offices, education, etc.). In particular, we have been able to monitor two locations in Madrid and two in Leganes, a smaller town nearby. In the following, we will refer to them as *Callao*, *Rastro*, *Leganes* and *IMDEA*. Overall, we collected more than 100 GB of LTE scheduling information, corresponding to a total amount of 8860 terabytes of transferred data in the four locations.

The city locations in Madrid are close to the city center and they are characterized by a high density of commercial activity, while the locations in Leganes are more residential. Although all four locations include both pedestrian and vehicular mobility patterns, the average users' speed in the city center is expected to be lower than that in Leganes. In all locations eNodeBs are placed on top of buildings of about four floors of height, but in Callao where the buildings are taller.

Table 10.1 provides statistics information of the four datasets. Although all the locations show a low median load (< 5 %), in all of them the load averaged over 5 minutes reached peaks as high as 70 % of the available resources.

Callao – The first measurement area is located in Madrid downtown, along the path of the most central shopping and restaurant street, Gran via. Figure 10.4 shows the sniffer and the eNodeB positions in a map of the surroundings.

The area surrounding the eNodeB location is one of the main squares of the city. Around it there are two cinemas (north and west of the map), one mall on the south-east corner and shops all around the are. The sniffer is located at the second floor of the building directly facing the square and without any direct obstacle between it and the eNodeB. The LTE signal received in this location belongs to Telefonica (Movistar), it is in the 1.8 GHz band and has a bandwidth of

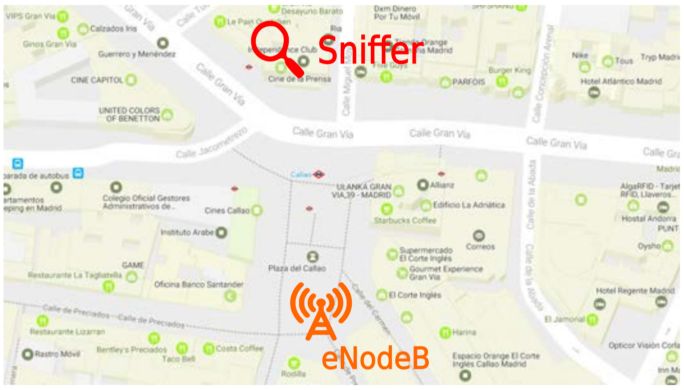


Figure 10.4: Callao area in Madrid downtown.

15 MHz. Mobile users in this area should be either pedestrian walking in the square or vehicular driving in the main street crossing the map from west to east. Given the location and usage of the area, the mobile network traffic should be concentrated between late morning and midnight, but some traffic should be present at any time.

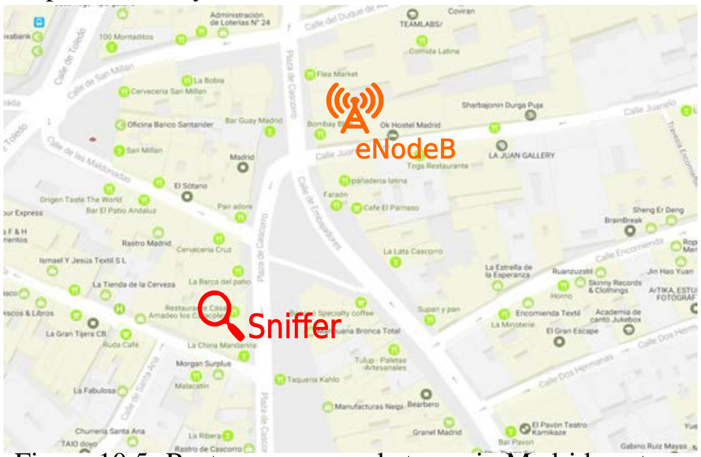


Figure 10.5: Rastro area, a market area in Madrid center.

Rastro – The second location is a market area of Madrid and takes its name from “el rastro”, the most famous flea market of the city. This is still a central area, but not quite as crowded as Callao. Figure 10.5 provides information about sniffer and eNodeB locations and the commercial activities in the surroundings.

The eNodeB location is on the roof of a short building on the corner of the crossroad, while the sniffer is placed in the third floor of an apartment in the nearby square. Although there is no direct line of sight between the two, the sniffer obtained a sufficiently high signal strength to decode the control channel. In the surrounding of the eNodeB, most of the commercial activities are either restaurants or small shops. The central area between the sniffer and the eNodeB is the market square.

The market is held every Sunday from the early morning to about 4 PM. The crowd in the surroundings is mainly pedestrian or slowly moving vehicles (mainly in the diagonal street going from west to north). The LTE signal received in this location belongs to Vodafone, it is in the 800 MHz band and has a bandwidth of 10 MHz.

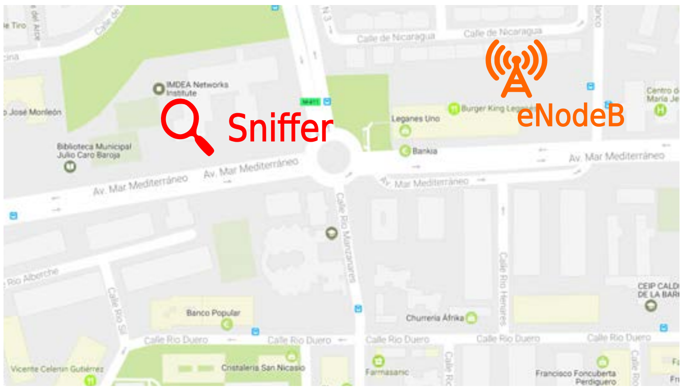


Figure 10.6: The area surrounding IMDEA in Leganes.

IMDEA – The third location is in Leganes, a town nearby Madrid, where our research center is placed. Figure 10.6 provides information about sniffer and eNodeB locations.

The eNodeB location is on the roof of a building within a green residential area. We placed the sniffer in one of IMDEA's meeting rooms so that the LTE antennas could be in line of sight of the sniffer. As a consequence, we obtained here a very high received signal strength. The area is mainly residential, but in the surrounding there are an elementary school, a small mall, a public office, a bank and a few small commercial activities.

Since the area is mainly residential, we expect the majority of the people to use private WiFi communication. The same is true for the public office employees. However, we expect a higher level of traffic during commuting time and when parents drives children to school. The LTE signal received in this location belongs to Vodafone, it is in the 800 MHz band and has a bandwidth of 10 MHz.

Leganes – The fourth and last location is in the center of Leganes. Figure 10.7 provides information about sniffer and eNodeB locations.

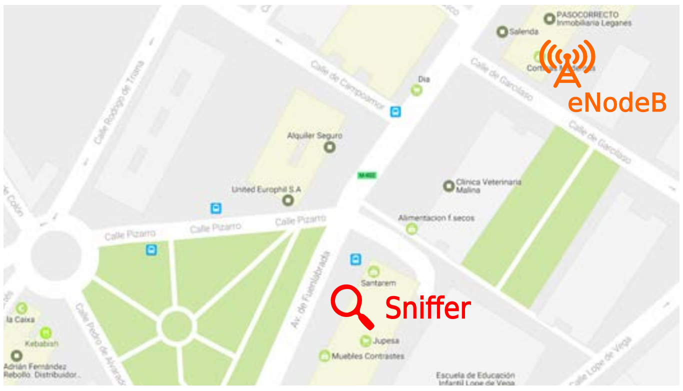


Figure 10.7: The sniffer and eNodeB locations in Leganes downtown.

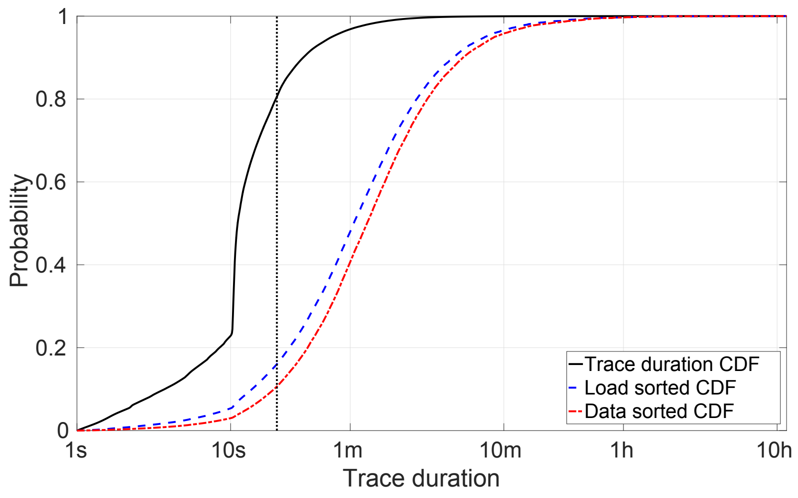


Figure 10.8: CDF of the trace duration mapped to the sorted CDF of load and data rate in Callao.

The eNodeB location is on the roof of a building facing the main street of the area. We could place the sniffer behind the window of a balcony in the direction of the eNodeB. A direct line of sight was not available, though, and the signal strength was just above the needed requirements for decoding the signal. Also this area is mainly residential with a few commercial activities in the surroundings. The main street is usually busy, but almost never jammed. The LTE signal received in this location belongs to Yoigo, it is in the 1.8 GHz band and has a bandwidth of 10 MHz.

10.3.2. Dataset Analysis

Since a user maintains her RNTI as long as she is active with no pause longer than 10 seconds, we split the traces accordingly: whenever a gap of 10 seconds or longer is present in a trace, it is split in two parts. Thus, we can analyze each trace in isolation and collect statistics about users network usage. In particular, each trace is a list of scheduling events concerning a particular user a containing:

- absolute time in milliseconds (LTE TTI)
- communication direction (downlink or uplink)
- MCS $\in [0, 31]$ (related to channel quality)
- N_{RB} (the number of resource blocks)
- transport block size (number of bits transferred)

For each collected trace we compute a set of compound metrics. The first three of them are trace duration, downlink trace size and uplink trace size. We first note that more than 60% of the collected traces are shorter than 10 seconds and are smaller than 10 kbit in terms of transferred

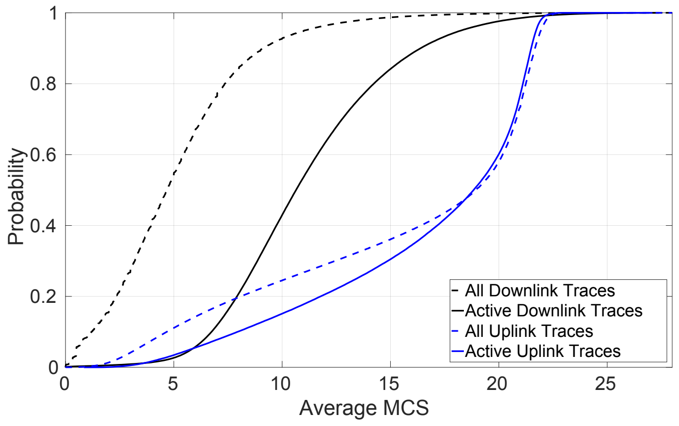


Figure 10.9: Comparison between MCS for active and all users for both downlink and uplink.

data. This means that the majority of the collected traces carries little or no information. We assume that these (small) traces belong to background traffic performed by mobile phones without any active intervention from the user or it is related to automatic network management operations.

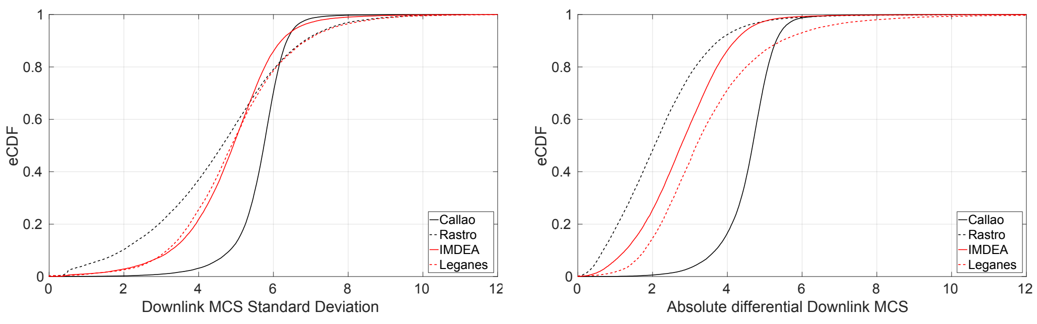
We analyze this in more details by computing the contribution to the total load of the traces longer than a given threshold or traces that transferred more than a given size of information. Figure 10.8 shows the trace duration CDF as a black solid line and maps the CDFs of the users' downlink load and total transferred size to their trace duration as dashed blue and dash-dotted red lines, respectively, for the Callao dataset.

The two CDFs represent the total load and data rate for all those users whose trace is longer than the value on the x-axis or, in other words, for a given duration on the x-axis, the three curves represent the fraction of traces shorter than that and the corresponding fractions of the total load and the total data transferred, respectively. Thus, traces shorter than 20s (dotted vertical line), which account for about three quarters of the total traces (black line) constitute 20% of the total traffic (blue dashed line). A similar behavior can be found when analyzing the transferred size compared to the total load and it is valid for both downlink and uplink and for all the datasets.

Our next consideration is that short or small traces are not relevant to the objectives of anticipatory networking optimization: in fact, they provide small chances for Quality-of-Service (QoS) improvements, because they introduce little traffic and they are difficult to predict due to their short length and, thus, difficult to be modeled. An additional evidence of this is obtained from the statistics of the average MCS measured over the traces.

Figure 10.9 shows the CDF of downlink (black) and uplink (blue) average MCS for all (dashed) and active (solid) users. Here we define a user to be active if its trace is either longer than 20 seconds or the transferred data size (either downlink or uplink) is larger than 100 Kbit. Note that this size corresponds to the size of a thumbnail image or that of a messaging application.

Both downlink and uplink CDF show that active users have higher average MCS, but also that downlink and uplink MCS distributions are quite different. The higher average MCS of active users is relevant for our analysis and shows that it is more likely for a user to be scheduled if



(a) (b)

Figure 10.10: Downlink trace variability characterization: on the left the CDF of standard deviation of the MCS used in active traces; on the right the absolute variation of the MCS.

she has a better signal quality, in case a larger volume of traffic is transmitted. However, the difference between downlink and uplink distributions, even though interesting per se, it is not directly relevant to the evaluation of anticipatory optimization. In fact, we believe they are mainly due to frequency division duplex: LTE networks have the uplink band at a lower frequency range to allow mobile terminal to save energy and this also provides a better signal quality due to the lower path loss.

Now that we defined active users/traces and their contributions, we address cell aggregated results computed for all users compared to the contribution of active users only. A user's achievable rate is a function of the assigned MCS, which is, in turn, a function of the path loss (i.e., Channel Quality Indicator (CQI)) and the error probability. Before evaluating the performance of prediction techniques on the collected traces, we analyze the MCS statistics and their variation over time. In particular, we evaluate for each active user, the following metrics: average MCS, median MCS, MCS standard deviation, MCS range, standard deviation of the binned average MCS, average binned standard deviation of the MCS, average absolute variation of the binned MCS.

While the first four metrics are standard statistics obtained on the whole trace, the last three metrics are obtained by evaluating the traces over bins of equal duration: for each bin of a trace we computed the average MCS and its standard deviation. The overall idea is that the average MCS should be linked to the average path loss/signal quality experienced by the user, while the standard deviation should be linked to fast signal quality variation (i.e., fading). Thus, evaluating these metrics over the whole trace and over bins, we characterize traces in terms of signal quality, noisiness and their variation over time. Ideally, for a trace to be easily predictable, it should have a low noisiness and low quality variation. Figure 10.10 shows the CDF of the MCS standard deviation in the four datasets, in the center, and the CDF of the average absolute variation of the binned MCS, on the right. In particular, Figure 10.10(a) shows that trace noise has a standard deviation usually smaller than 6 which means the range of MCS variation is small compared to the maximum range of 28. Also, the Callao dataset shows the highest noise, which can be a

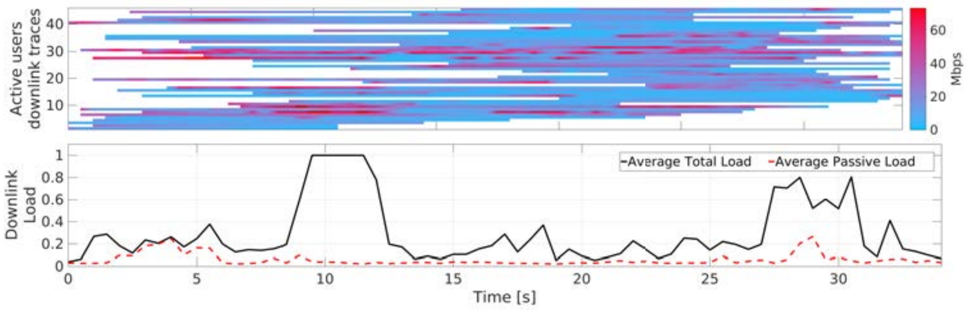


Figure 10.11: A 35-second portion of the downlink channel of the Callao dataset. Each row of the top chart shows the MCS evolutions of an active user. The lower chart provide aggregated information of the cell traffic.

consequence of the particular topology of the area. Figure 10.10(b), which measures how fast the MCS varies in subsequent bins, tells us that the traces in the dataset have a slow to medium dynamic with successive MCS changes around 2-3 (max. range 28), which means that rapid large variations in MCS are not common.

10.4. Evaluation and discussion

In this section we investigate the performance of the different optimization approaches and degrees of prediction accuracy. To evaluate our framework, we proceed by selecting small portion of the datasets. Figure 10.11 provides an example of a 35-second analysis of the downlink channel, containing 45 active users. The top chart shows the evolution of the MCS for all the active users in the time frame, where each users is represented by a separate row and the color varies from white (no communication), to light blue (bad channel quality, few Kbps) fading into red (good channel quality, tens of Mbps). The bottom chart, instead shows aggregate information about the cell traffic: the average total load is shown as a solid black line and the contribution to the load generated by background traffic as a dashed red line.

Each portion of the dataset is generated as follows:

- select a subset of the dataset of length T and starting at time τ
- identify all N active users in the subset and retrieve their MCS traces
- create the ground truth elements $r_{i,j}$ from the MCS traces using the tables in the standard [220] to compute the transport block size for the maximum number of resource blocks. The ground truth is created for $i \in [1, N]$ and $j \in [\tau - \Delta_T, T + \Delta_T]$, where Δ_T is a margin to remove boundary effects from the evaluation.
- create ARIMA models and proactive predictions for all N users
- create minimum requirements $d_{i,j}$ and used resources $a_{i,0}$ as the amount of exchanged traffic and used resources, respectively

- create the background load $a_{B,j}$ for $j \in [\tau - \Delta_T, \tau + T + \Delta_T]$ summing the load of all non-active users
- run all the optimization schemes and compute their performance on the central time span $j \in [\tau, \tau + T]$. We refer to the resource allocation computed by the optimizer as $a^*_{i,j}$.

Thus, for each analyzed time span we obtain the *resource saving* percentage as

$$\Delta_a = \frac{100}{N} \sum_{i=1}^{N} \left(1 - \sum_{j=\tau}^{\tau+T} a^*_{i,j} / a_{i,0} \right), \tag{10.5}$$

the *data rate increase* percentage as

$$\Delta_r = \frac{100}{N} \sum_{i=1}^{N} \left(\sum_{j=\tau}^{\tau+T} a^*_{i,j} r_{i,j} / d_{i,0} - 1 \right) \tag{10.6}$$

and the *total outage* as

$$L = \sum_{i=1}^{N} \sum_{j=\tau}^{\tau+T} l^*_{i,j}. \tag{10.7}$$

Due to the intrinsic computational complexity of the problem that entails training ARIMA predictors and solving several multi-objective LP systems, it was not feasible to apply a brute force method to evaluate the results over every single portion of the four datasets. Instead, we opted for exploring the datasets in order to cover their characteristic uniformly. During the aggregated information analysis, we also associated each analyzed dataset portion to its average characteristics (e.g., load, MCS statistics and prediction MSE). Then, we computed the statistic distributions of these characteristics in the datasets to obtain bins such that the same fraction of the total load falls in each of them. Finally, we select the next portion of the dataset to analyze from the bin that contains the fewest samples. In such a way, we can assess the impact of the different features of the dataset on the performance of the anticipatory networking techniques.

In addition, in order to apply anticipatory networking optimization we assume that each active user's traffic can be re-organized as if it is generated by a multimedia streaming application: future data transfers can be buffered as soon as the trace started and up to the maximum buffer size. When not specify otherwise, the buffer is assumed to be infinite. Finally, while ideal methods are computed at once on each analyzed portion of the dataset, realistic methods are iteratively updated in each time slot to recompute predictions and re-evaluate the solution of the optimization framework.

We start with the performance of the ideal resource minimization optimizer with perfect future knowledge over a whole day. Figure 10.12 illustrates as a solid black line the average resource percentage saved over 30-minute moving windows. Grey dots represents single results computed over time spans of $T = 10$ and $\Delta_T = 5$ seconds. The blue dashed line illustrates the cell load

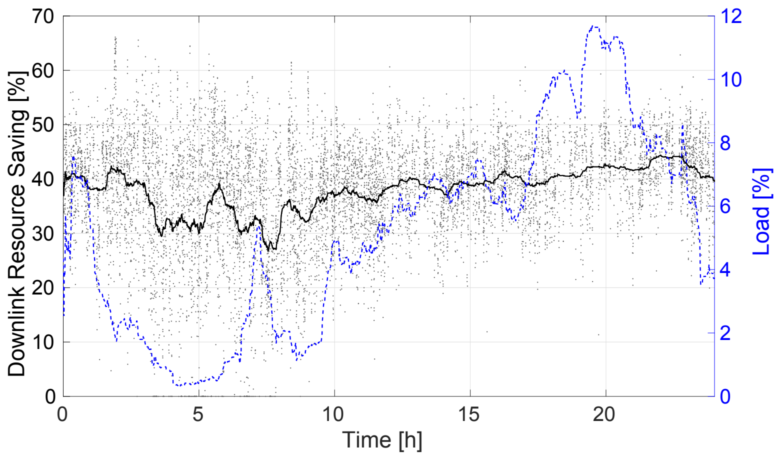
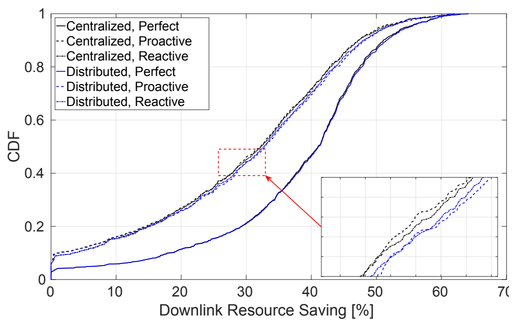


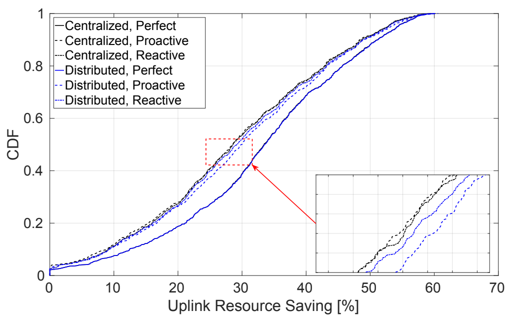
Figure 10.12: Variation of the ideal optimizer performance over a full day downlink traffic in Callao, compared to the cell load. Lines illustrate the moving averages of the parameters, while dots are single results.

variation averaged over 30-minute moving windows. The figure is obtained for the downlink channel of the Callao dataset.

The average performance of the resource minimization solution is very good. In fact, the solution is able to maintain an average saving almost always higher than 30% and up to 45%. However, the instantaneous performance of the solution is much more variable and spans the whole possible range from 0% (no improvement) to about 65%. These extreme conditions happen more frequently when the load of the cell is very low and, thus, they are symptoms of critical conditions in the analyzed portion of the dataset: such as a single active user whose trace is either already optimal (for 0%) or it allows for very high saving ($> 55\%$). For what concerns the impact of the cell load on the optimization performance, we cannot determine any strong correlation by visual inspection. However, the range of individual results is wider for low load, while it gets smaller when the load is higher. We believe that when the cell load is higher, there are also more active users in the cell and, thus, the overall characteristic tends towards the average condition of the cell, while when the load is low, the individual behavior of each user dominates the aggregate characteristic of the cell traffic and determines the system performance. Figure 10.13 shows the CDFs of the resource saving performance obtained by the three prediction accuracy levels (perfect, proactive and reactive). The strongest impact on the system optimization is caused by replacing the perfect knowledge by more realistic approaches. Also, the chosen realistic approach does not strongly affect the amount of saved resource. A close inspection (see the zooms in the lower right part of the figures) allows to see the difference between the reactive and proactive predictions. Although they fare very similarly, the figures show that some higher resource savings are obtained by the reactive approach. This result might seem counter-intuitive, but is justified examining the other KPI: the outage time. In fact, while the proactive scheme never suffers from any outage, the reactive does and, although the amount of outage is always smaller than a single time slot, it is sufficient to allow the optimizer to achieve some extra resource savings.



(a) Downlink



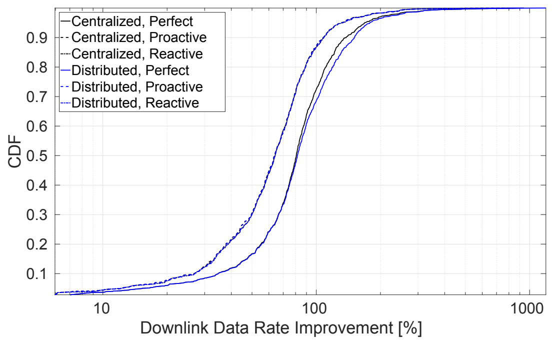
(b) Uplink

Figure 10.13: CDFs of the resource saving obtained by anticipatory networking solutions for different prediction accuracies.

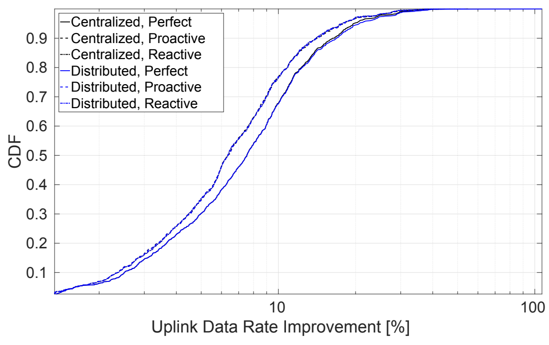
Overall the performance degradation due to realistic prediction methods ranges from 5-10% for high savings ($> 40\%$), to 10-15% for moderate savings (20-40%) to more than 15% for low savings. Even though this last condition happens in fewer than 15% of the analyzed cases, these are the cases where anticipatory networking is more likely to be useless or detrimental to the users' QoS: in fact, while some resources are still saved, they might be saved at the expenses of some outage, which will impact the users' experience.

The performance of centralized and distributed optimization schemes (black and blue lines, respectively) do not show substantial differences. Moreover, when they differ the distributed variants perform slightly better. To understand these two counterintuitive results, we analyze the achievable rate traces of the active users and the resource allocations obtained by the two schemes. Both schemes assign resources to a given user by prioritizing the time slots with higher achievable rate, however, the centralized scheme considers all users at the same time, while the distributed version optimize each user separately. In order for the distributed solver to have the same performance of the centralized version, the resource allocations obtained for each user must be *compatible*. We call compatible a set of allocations that can be superimposed without creating any unfeasible condition (i.e., requiring more resources than those available in a given time slot). All the test cases analyzed in details showed one of the two following outcomes. The first and more common situation has all users to have achievable rate peaks in different time slots so that their optimized allocations do not collide or, if they do, their combination does not exceed the available resources. The second situation, which is more rarely verified, has two or more users showing simultaneous peaks of achievable rate and, thus, the resource allocations computed by the distributed solver collide in one or more time slots. These collisions reduce the amount of resources assigned to all users so that the service is momentarily interrupted (i.e., outage). Since the distributed scheme trades some outage for some lower resource utilization[1], the performance of the centralized scheme seems worse.

[1]We do not include graphs for the outage KPI, because they were consistently very low for all schemes, but the centralized optimizer with perfect prediction for which they were exactly zero.



(a) Downlink



(b) Uplink

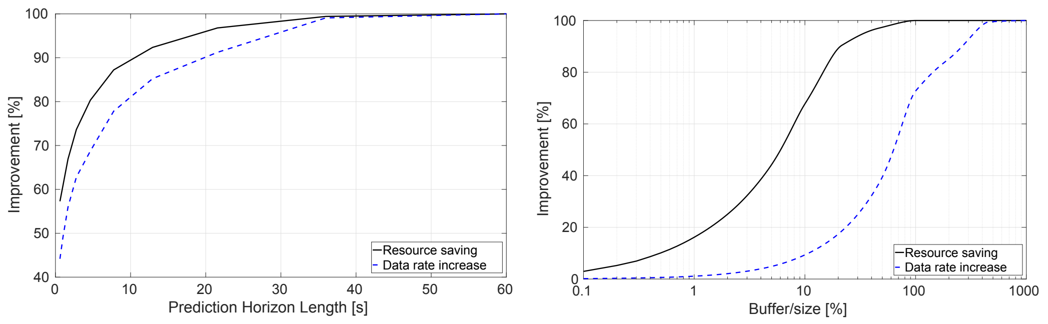
Figure 10.14: CDFs of the data rate increase obtained by anticipatory networking solutions for different prediction accuracies.

Figure 10.14 shows the CDF of the rate maximization performance and is equivalent to the previous in all aspects, but for the magnitude of the improvements. In fact, the rate maximization solutions are able to more than double the data rate for the downlink channel. Conversely, in the uplink the improvements barely reach 40%. This disparity of performance is justified by the different MCS statistics of the downlink and uplink channels, of which the second is consistently higher. In turn, this translate into a smaller margin of optimization for the uplink data rates, see Figure 10.10(b) for a comparison of the MCS CDF and Figure 10.11 for a detailed representation of MCS traces in both channels, where all uplink traces are represented with lighter shades of gray (i.e., higher MCS). Overall, we measured data rate improvements between 20% and 100% with a median value of 65% for downlink communications and between 3% and 13% (median 6.5%) for the uplink. Curves for centralized and distributed show again that the two schemes fare very similarly. The reasons are the same described in the previous paragraph.

We also compared the CDFs of the two main KPIs computed in each dataset separately: the performance computed by our optimization framework does not differ by more than 5-10%, but for the Leganes dataset. We attribute this to the low load and mostly residential characteristics of this dataset.

To conclude this evaluation, we show in Figure 10.15(a) the impact of the prediction horizon. Basically, the prediction horizon represent the number of time slots optimized at the same time. Thus, a shorter horizon makes the optimizer less effective as it can only rely of short term information. In the figure we show normalized average results in order to be able to compare solutions with different performance. The chosen examples consider a maximum prediction horizon of one minute and analyze the same by giving the optimizer a fraction of the whole available information. Although the best performance is reached asymptotically, substantial improvements can be obtained with just a few seconds of prediction.

This graph helps understanding why the realistic predictors performs so closely. In fact, reducing the prediction horizon of the omniscient predictor makes it similar to a realistic one



(a) Horizon (b) Buffer

Figure 10.15: Anticipatory networking performance varying the prediction horizon length (left), and the buffer size (right).

which is more effective in the first time slots only. As such we can compare a realistic (either proactive or reactive) predictor to an omniscient one with an horizon of about 10 seconds. Finally, for what concerns the impact of the buffer size on the optimization performance, Figure 10.15(b) shows the normalized average improvements for the two KPIs. The fraction of buffer given to each of the active users is proportional to the amount of requested data in the time frame. Thus, a 100% buffer would allow a user to prefetch everything at the beginning of the trace. While increasing the buffer size over 100% does not improve the resource saving, the data rate can be further increased by allowing the user to buffer more data. In particular, in our test conditions a buffer four time as large as the requested data transfer allows for maximum performance gain.

A few final considerations about the overall approach are in order. The first concerns our datasets: optimizing network resource allocation starting from real traces makes it impossible for the optimizer to run into infeasible conditions, because the starting point was already feasible. The second consideration concerns whether the anticipatory gains can be estimated from the trace characteristics without solving the optimization problem. Studying the correlation between our final results and the compound metrics computed above for active users we found they are almost independent. This is due to the fact that the degree of improvement does not depend on the characteristics of individual users, but on their combination. Determining whether combining different users results in a good mix and provides high gains is a problem just as complex as the resource allocation problem itself.

Chapter 11

Conclusions

In this thesis we investigated prediction-based techniques for the optimization of mobile networks. Our main objective has been to devise the theoretical gain achievable by feeding optimization solutions with knowledge of the future system evolution and to investigate whether such a system was feasible. The whole material have been split in four parts: an introductory part that reviewed anticipatory networking solutions, a second theoretical part where we proposed three novel optimization solutions that considered different optimization objectives, a third practical part where we described the tools and the evaluation methodologies we developed and a fourth conclusive part, which presented a thorough evaluation of the proposed solution on our large data set of LTE scheduling information.

In details, Chapter 2 comprehensively reviewed the state-of-the-art on anticipatory networking [3]. This survey included a thorough study on applications exploiting a variety of type of contextual information to build prediction framework that are in turn used to drive optimization solutions. Prediction and optimization techniques are further analyzed in the two following sections. There we identified the most popular approaches in the literature in order to provide guidelines and best practices to select the best solution to deal with a given application or data set. We concluded the survey with a study on the open issues the have to be solved to successfully adopt the anticipatory networking paradigm in future generation networks. Chapter 3 and Chapter 4 described the two models we devised to describe the impact of prediction error on the estimation of the achievable rates in mobile cellular networks. The first model [4] proposed a two time-scale framework which describes short-term (e.g., tens of seconds) predictions with Gaussian process, and medium- long-term (e.g., one minute or longer) predictions with a convolution of statical distributions to account for geographic and congestion uncertainties. The second model [5] refined the short-term part of the previous one: we empirically find out that Gaussian random walks closely match the distribution of prediction errors.

The second part of this thesis consists of Chapters 5, 6 and 7, which described our original contributions in terms of prediction-based optimization techniques. The first of the three solutions [6] is the direct consequence of our two time-scale prediction error model applied to the theoretical optimal solution of the resource minimization problem for a single user case. We showed that our iterative approach is almost optimal in term of service outage and allow the network to save up to 30% of the resources. While the omniscient single user case can be solved very efficiently in linear time over the time span of the optimization, Chapter 6 extended the problem to multiple user and variable requirements. The complexity of new optimization problem proved to be exponential over the problem size [7]. Thus, we proposed an heuristic solution that asymptotically approaches the optimal solution, but can be stopped at any time, always providing a feasible solution to the multi-user data rate maximization problem. Analyzing the multi-user scenario highlighted that without controlling the number of admitted user it was impossible to enforce any given level of Quality-of-Service. Chapter 7 illustrated how to modify the multi-user data rate maximization problem in order to enforce QoS to the largest set of users [8]. As per the previous one, the complexity of this problem encouraged us to develop an iterative solution based on a fast LP formulation of the multi-user problem. All the solutions of this part have been validated on synthetic traces and showed that anticipatory networking solutions could offer data rate increase of about 50%, resource savings of about 30% and that, QoS enforcing was feasible.

To validate our work on real data we needed to develop a few missing tools and methodologies that are described in the third part of this thesis. Chapter 8 described the tool that provided us with the needed data set: a decoder of the LTE control channel that we called OWL [20]. LTE control channel broadcasts all the scheduling information related to every user allocation in both the downlink and the uplink channels and, since it is unencrypted, our tool can decode and log how many resources are assigned to any user and the modulation and coding scheme used in that communication. Our tool is the main option available to the research community[1] to access information that were only accessible to operators or by buying expensive commercial equipments. Chapter 9 detailed how we used OWL to devise how accurate is the estimation of data link achievable rates computed by mobile phones. Our measurement campaign on three commercial phones [256] showed that mobile phone measurements can achieve an accuracy close to 95% and a precision higher than 90% even on short-lived communications (e.g. 50 ms or 100 KBytes).

The fourth part of this thesis described the final real world evaluation we carried out using the data collected with our tool [9, 10]. Having recorded the LTE scheduling information in four different locations in Madrid for a month, we could assess the performance of our prediction-based optimization techniques on real data. Chapter 10 described our measurement campaign, the evaluation framework we devised to study the 100 GB of collected scheduling information (more than 8 thousands of TeraBytes of exchanged data) and the obtained results. We observed that the resource savings obtained by our framework reached 65% in the best scenarios and were between 30% and 40% on average. We also studied the achievable data rate increase if the total amount of used network resource were maintained: in the downlink our techniques could almost double the offered data rate, while in the uplink the gain was only about 7%, because of the

[1]OWL source code is available at `https://git.networks.imdea.org/nicola_bui/imdeaowl`.

smaller MCS margin in these communications. The performance of anticipatory optimization on real-data were slightly better than what obtained on synthetic data. This is mainly due to the randomness of synthetic traces that did not re-create the same level of variety that is present in real datasets.

Our final evaluation of the impact of prediction on optimization techniques for mobile networks is that it can play an important role in 5G and future generation networks. In fact, by enabling substantial resource savings and data rate improvements it can be one of the main enablers of future high data-rate mobile applications. In addition, this work paves the way for multiple new directions: our measurement tool is already being used in other groups to study, for instance, energy savings, communication quality, and traffic profiles. In addition, by combining our framework with machine learning techniques it will be possible to study application recognition solution and identify malicious behaviors by anomalous traffic fingerprints. At the same time, to develop a real anticipatory networking framework for mobile networks, the major missing part is related to user data collection, storage and management. These three features needs to be addressed both from the technological and from the legislative points of view. Finally, we believe our work demonstrated that anticipatory optimization is worth being adopted and standardized for the next generation of mobile networks.

Appendices

Appendix A

Lightweight Mobile Bandwidth Availability Measurement

This chapter proposes a simple technique which is able to measure the fast variations of the per user capacity and, from those, the expected end-to-end throughput. In order to do so we adapt packet train dispersion techniques by applying an adaptive filtering mechanism, which we show is effective in removing the impact of outliers due to bursty arrival and jitter, which are very prevalent in mobile environments. We validate the effectiveness of the solution through extensive simulation and measurement campaigns: our technique can achieve an accurate throughput estimate with as few as 5 % of the packets needed by other solutions, while making an error smaller than 20 %.

This enables filter based prediction techniques and, consequently, prediction based resource allocation optimization.

The rest of the chapter is structured as follows. Related work is discussed in Sections A.1, the measurement technique in Section A.2, a first evaluation in Section A.3 and the measurement in Section A.4. The results are discussed in Section A.5.

A.1. Related Work

A number of approaches exist to estimate mobile capacity. The most popular of which is Ookla's mobile application, Speedtest [234], which computes the maximum end-to-end throughput achievable by two long lived TCP connections with the closest measurement server (according to our tests the measurement lasts for either 20 seconds or after 30 MB have been downloaded, whichever happens first). Then, it derives throughput samples and aggregates them into 20 bins (each one has about 5% of the samples), applies some post processing to remove measurement artifacts and, finally, estimates the average of the bins. Huang et al. [229] proposed to use 3 parallel TCP connections in order to remove the effects of packet losses, TCP receive window limitations and overloaded servers, while ignoring any data collected during the slow-start phase of TCP.

The calculated throughput is given by the median of the collected samples, in order to reduce the effect of outliers. Recently, Xu et al. [235] analyzed the use of User Datagram Protocol (UDP) to compute the end-to-end throughput availability, also accounting for packet interarrival times and the impact of mobile scheduling. All these techniques are active, use long data transfers and thus, incur a high overhead.

Conversely, passive monitoring techniques aim at estimating similar information by analyzing ongoing mobile communications, without triggering any dedicated activity. Gerber et al. [257] achieved quite accurate results just by relying on selected types of applications (i.e., video streaming), which provide more reliable throughput measurements as they are more likely to exploit the full cell capacity. In order to study transport protocols in LTE, [230] developed a passive measurement scheme, which monitors the sending rate over a given time window that ensures the full exploitation of the capacity. PROTEUS [113] combines passive monitoring with linear prediction to estimate the achievable throughput. Other solutions worth mentioning in this category are [258], where the authors try to identify bottleneck links in the core network of an operator by conducting large scale passive measurements of TCP performance parameters and [259], where network "footprints" (generated by counting the number of packets and the number of retransmissions of all the users of a network) were used to identify capacity bottlenecks. However, these solutions cannot be directly applied to mobile phones. We conclude that none of the aforementioned solutions allow for frequent throughput measurements, nor do they provide estimates of the per user cell capacity on the client side (mobile device) to allow for effective capacity prediction and resource allocation.

Lai [260] attempts to actively measure the link capacity (which in [260] is called bandwidth) of a path by taking advantage of the packet pair property of FIFO-queuing networks. Dovrolis [261] further refines the packet pair technique and demonstrates that packet pair dispersion rate has a multimodal distribution, whose modes in turn depend on the capacity and the cross traffic at each of the links composing the sender-receiver path. Also, the authors devise a method to estimate the capacity of the bottleneck link in the path, based on the fact that the average throughput measured by packet trains converges to the asymptotic dispersion rate, from which an estimate of the bottleneck capacity can be computed. As we will discuss later though, it is unsuitable for use over mobile networks. CapProbe [262] proposed a technique based on packet pairs dispersion and delays to devise a reliable capacity estimation technique, aimed at mobile networks. Both techniques are meant to measure the capacity of the bottleneck link of a path. Instead, we are interested in measuring the per user capacity at a given moment.

A.2. Mobile Capacity Estimation

In the literature, the term *link capacity* refers to the transmission rate of a link, *path capacity* is the minimum transmission rate among all the links of the path and finally *link available bandwidth* refers to the spare link capacity (capacity not used by other traffic) [261]. Instead, we are

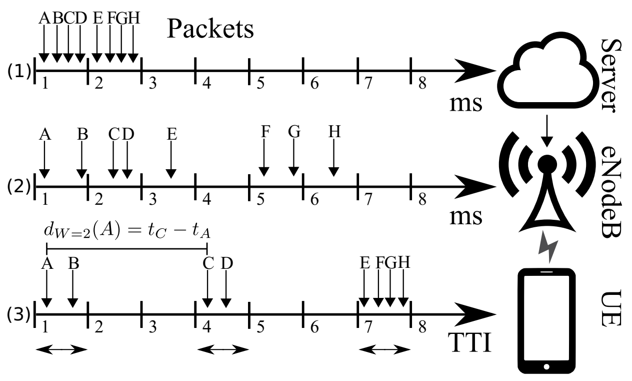


Figure A.1: Dispersion of IP packets over the Internet. First, they are sent back-to-back from the server (1). After experiencing dispersion on the Internet, they arrive on the BS (eNodeB) (2). Finally, they are received in groups by the UE (3). The timelines (1-3) in the figure happen sequentially, one after the other, not in parallel. The horizontal arrows represent a TB allocated to the recipient UE.

interested in estimating the maximum capacity that the scheduler of an eNodeB could allocate to a target user if he requested saturation traffic under a specific bearer. This metric is specific to cellular networks, we call it *per user capacity* and we symbolize it as C_U. For brevity, in the rest of the paper we refer to it as *capacity*. To the best of our knowledge, traffic flow templates are not used for generic browsing and multimedia traffic, which is the scope of this work. Thus, we can safely assume that all the measured traffic is using the default bearer, allowing us to ignore this variable. As we will analyze in the sequel, in practice, the measured C_U will often be less than the maximum capacity a user could be allocated. For this reason, the measured value represents the greatest lower bound of the user's capacity. We will show that this value is very close to the actual maximum, thus causing a slight underestimation of the true maximum per user capacity.

The wireless link is the last hop of a downlink path and the C_U of all the connected users is dependent on the cell congestion, the channel quality, the channel's bandwidth and the scheduling algorithm. It is usually the link of a path with the lowest capacity, that also contributes the most to the delay. On the other hand, the average end-to-end TCP throughput R, depends on the capacities and the cross traffic of all the links in the path, as well as possible rate adaptations at the server side, caused by the TCP mechanisms. The end-to-end TCP throughput is primarily determined by the link with the minimum spare link capacity, which in a mobile scenario is usually the Radio Access Network (RAN). We are interested in measuring C_U, since it is the metric that affects all the connections that the user is going to have in the future and is usually the bottleneck.

Figure A.1 illustrates the packet dispersion due to the transmission over links at different link capacities. This example is based on LTE, but similar effects are observed in various mobile technologies. Initially, (1) the server sends a burst of Internet Protocol (IP) packets (A-H in the example) back to back. The number of packets in the burst varies since it depends on a number of factors like the state of TCP connection, the specifics of the application and the server that generates it. Subsequently, (2) the base station (eNodeB) receives the packets, which have suffered variable delays due to the different link capacities and cross traffic encountered along the path. When the scheduler allocates a TB (marked with horizontal arrows in the plot) to the receiving UE (3), as many packets as possible are encapsulated in it. Therefore, all the packets that are scheduled together arrive within the same TTI at the UE. As a consequence, the inter-packet interval can be greatly reduced (packets A and B) or greatly magnified (packets B and C).

Considering the set of back-to-back transmitted packets crossing the path in Figure A.1, we can distinguish their arrival rate R_A at the antenna from their transmission rate from the antenna to the user, which can have a maximum value of C_U. Both metrics are dynamic and are affected by the same parameters that affect R. Thus, if we sample them for a specific period of time, we may notice the following relationship between them. If $R_A > C_U$, the set of packets arrives at the BS with a delay which is inversely proportional to R_A and shorter than the average time needed for the BS to serve all but the last packet. Since the arrival rate is higher than the departing rate at the base station, the dispersion of the set is caused by the last link. Also, depending on the scheduling strategy, the set may be served within the same transport block or multiple transport blocks by the BS. Conversely, if $R_A < C_U$ the set of packets arrives at the BS separated by a delay which is longer than the average serving time of the BS. We thus have three cases: *i*) bursty arrival [230, 235] (e.g.: set of packets E-F), if $C_B > C_U$ and packets are in the same transport block, *ii*) last hop capacity if $C_B > C_U$ and packets are in different transport blocks (e.g.: set of packets A-D), or *iii*) lowest hop capacity if $C_B < C_U$.

In order to estimate C_U, we have to filter both *i*) and *iii*) cases, as well as take into account the behavior of sets of packets when transmitted over mobile networks. In brief, our approach has two components: a) generating capacity estimation samples which are not significantly affected by the above and b) the statistical processing of those samples in order obtain a C_U value.

A.2.1. Capacity Estimation Samples

The input data for our passive measurement tool are the timestamps and sizes of all the received data packets of a smartphone. We ignore packets related to connections establishment such as TCP and Transport Layer Security (TLS) handshakes, since they can not saturate even momentarily the wireless link. This information can be collected on the Operating System (OS) level by monitoring the stack. In our experiments, we use rooted Android smartphones and tcpdump to capture all the incoming traffic. Ultimately this functionality could be included in the mobile OS as an on-demand lightweight measurement service.

We consider a set of N packets sent from a server and received at the UE so that the i-th packet is received at time t_i, with $i = \{1, \ldots, N\}$. A key metric used by our algorithm is the *inter-packet interval*, the time difference between the arrival of two consecutive packets ($t_{i+1} - t_i$). Obviously, in a group containing N packets, there are $N - 1$ intervals. W represents the unit-less number of such intervals that we take into account when we generate the capacity estimation samples. For each packet in the set we define the dispersion time $d_W(i) = t_{i+W} - t_i$, and the per user capacity

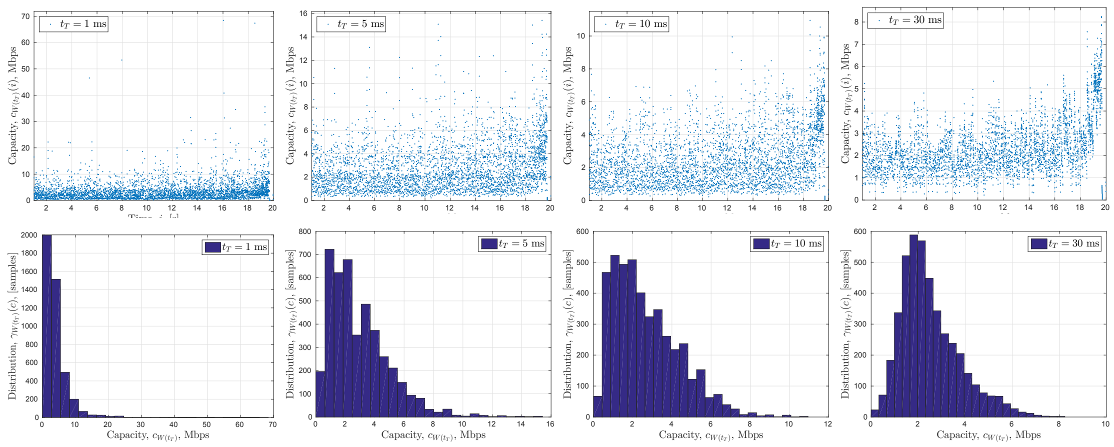


Figure A.2: Scatterplots of $c_{\hat{W}}$ (top row) and histograms of $\gamma_{\hat{W}}$ (bottom row) computed for $t_T = \{1, 5, 10, 30\}$ ms from left to right. When the dispersion time is computed on windows larger than the TTI, $t_T > t_S$, the dispersion time distribution gets more stable.

sample $c_W(i) = (\sum_{j=i}^{i+W-1} L_j)/d_W(i)$, for a given value of W, where L_i is the length of i-th packet.

In detail, the $c_W(i)$ value of packet i is derived by adding the sizes of W consecutive packets, starting from i and then dividing by the time duration of W consecutive inter-packet intervals, starting from $[t_{i+1} - t_i]$. Packet $i + W$ contributes only to the denominator. For example, in Figure A.1, $c_{W=2}(A)$ is computed by dividing the sum of sizes of the packets A and B by the dispersion time $d_{W=2}(A) = t_C - t_A$.

The three arrival cases above contribute to the distribution of the capacity samples in different ways. Arrivals of type *i*) cause a tiny d_W and, thus, skew the distribution to the right (over-estimation of C_U). At the same time, type *iii*) events, which show larger d_W (under-estimation of C_U) skew the distribution towards the left. To better visualize what is discussed next, Figure A.2 shows a set of scatterplots of c_W and histograms of its distribution computed on a single download performed using the Speedtest application [234] over a High Speed Packet Access (HSPA) connection. The X-axis of the scatterplots represents the arrival time of packet i and the Y-axis its c_W value.

The impact of type *i*) arrivals can by limited by setting W appropriately. The idea is to include in each measurement packets belonging to different TBs in order to make sure that the highest throughput c_W we can measure is only related to the cell capacity and not to bursty packet arrivals, as it would have happened had we chosen $W = 1$ in the example of Figure A.1. In order to achieve that, it is sufficient to study groups that, starting from any packet i, contain W_i intervals so that the minimum dispersion time $d_W(i)$ is longer than the maximum TTI of the scheduler, abbreviated t_S:

$$W_i = \{\min(W) \mid \min_W(d_W(i)) > t_S\} \tag{A.1}$$

This guarantees that at least two packets within the W_i window are scheduled in two different transport blocks, since $t_{i+W_i} - t_i = d_{W_i}(i) > t_S$. In other words, we are averaging the burstiness over two transport blocks. An effect of Equation (A.1) is that each packet i has a different W_i value, depending on the spacing of packets that were received after it.

It is important to select the minimum value of W for the creation of the $c_{W_i(i)}$ value for packet i that has the property $\min(d_{W_i}(i)) > t_S$. The "slow start" behavior of TCP introduces noticeable gaps in packet delivery. Thus, samples that include these gaps in their calculation of d_W, generate c_W values that are significantly smaller and not representative of the C_U. A high value of W increases the probability of a sample to include such gaps.

A.2.2. Statistical Processing Of The Samples

Now that type *i*) events are filtered, we ensure that each set spans across at least two TBs. The minimum dispersion time $\min d_{W_i}(i)$ for every packet i of the flow cannot be smaller than the minimum time needed for a set of packets to cross the wireless link, which corresponds to the maximum per user cell capacity. Thus, C_U can be found as the maximum of the distribution of c_W, which is equivalent to the maximum value of c_W.

$$C_U = \max_{i \in [1,\ldots,P]} c_{W_i}(i) \tag{A.2}$$

P is the total number of data packets of a flow. Note that, with Equation (A.1) we are filtering the effect of type *i*) arrivals $(\min)$ and with Equation (A.2) the delays introduced by type *iii*) arrivals $(\max)$.

Ideally, we would like to sample c_W until its distribution is stable, but C_U is varying because of both user movements and fast fading. Hence we can only obtain an estimate ${C_U}^{(p)}$ of it from a set of p consecutive estimation samples, where $p < P$. Although estimating the distribution from a limited number of samples reduces the accuracy of our measurement, we can at least guarantee that we are not overestimating C_U:

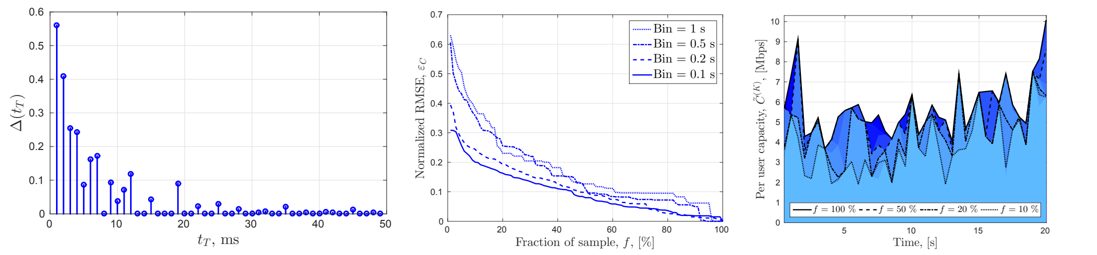
$${C_U}^{(p)} = \max_{i \in [1,\ldots,p]} c_{W_i}(i) \leq \max_{i \in [1,\ldots,P]} c_{W_i}(i) = C_U \tag{A.3}$$

This follows from the probability of the distribution of a sampled random process to contain the maximum of the theoretical distribution of the process, which is increasing with the number of collected samples:

$$\lim_{p \to \infty} {C_U}^{(p)} = C_U \tag{A.4}$$

A.2.3. Capacity Measurement

This section describes the feasibility of lightweight active and passive measurements of per user capacity C_U based on dispersion samples of packet sets. It also explores the effect different values of some parameters have on our technique. We compute the dispersion time by using an



(a) Ratio $\Delta(t_T)$ (b) RMSE ε_C (c) Timelapse

Figure A.3: Left: Ratio $\Delta(t_T)$, varying $t_T \in [2, \ldots, 50]$ ms. The measurements get stable from $t_T > t_S = 10$ ms. Center: Normalized root mean square error ε_C of the capacity estimate computed over a fraction $f = K/N$ of continuous samples for varying bin sizes ({0.1s, 0.2s, 0.5s, 1s}). Right: Time plot of the capacity variation $C_U(t)$ computed every 500 ms and its different estimates computed with $f = \{10, 20, 50, 100\}$ %.

adaptive window W_i intervals long for every packet i such that:

$$W_i = \{\min(W) \mid t_{i+W} - t_i > t_T\}, \tag{A.5}$$

where $t_T \in [1, \ldots, 50]$ ms, for all the values of t_T. The estimation sample of the i^{th} packet is composed of all packets following i until the first packet which arrived at least t_T ms later than i. This allows to satisfy Equation (A.1) a posteriori if the TTI duration is not known.

We exemplify the dispersion time in Figure A.2 based on data obtained by time-stamping the arrival time of the packets of a 6 MB HSPA download. The figure presents the evolution of the scatterplots of c_W and the corresponding histograms of the c_W distribution for various characteristic values of t_T.

During the slow start phase of a TCP connection an increasing number of packets are sent back to back from the server, and after a few RTTs the congestion window is large enough to allow the transmission of packet trains long enough to measure capacity as high as 100 Mbps. In fact, C_U should be proportional to the maximum number of packets that can be scheduled in a single transport block and, if Equation (A.1) is satisfied and $t_T > t_S$, the impact of outliers due to bursty arrivals is removed. With reference to Figure A.2, it can be seen that the maximum of c_W is approaching a stable value of about 10 Mbps when $t_T \geq 15$ ms. Due to limited space, we do not present the related plots of other downloads. Based on the rest of our dataset, a stable value is reached for values of t_T between 10 and 20 ms.

Moreover, Figure A.3(a) shows the stability of the maximum of the capacity by plotting the ratio $\Delta(t_T)$, computed between the maximum value obtained with windows of $[t_T]$ and $[t_T - 1]$:

$$\Delta(t_T) = \frac{|C_{W|t_T} - C_{W|t_T-1}|}{C_{W|t_T-1}} \tag{A.6}$$

Ideally, the ratio $\Delta(t_T)$ should stabilize to 0 as soon the scheduling outliers are filtered ($t_T >$

t_S) and further increasing t_T should only make the distribution smoother. However, in actual experiments increasing t_T makes it more difficult to obtain a sample of the maximum capacity which is consistent over different transport blocks. In this preliminary example, we can see that $\Delta(t_T)$ becomes stable for $t_T > 20$ ms, which is in line with the HSPA TTI of $2 - 10$ ms.

Next, we divide the time duration of a download into fixed sized bins. We apply the above method taking into account only a percentage $f = k/K$ of consecutive capacity samples in each bin. In this case, K is the total number of samples inside each bin and k is the number of consecutive samples that we consider for every bin. Figure A.3(b) shows the coefficient of variation of the normalized root mean square error – CV(NRMSE) – of the estimate ε_C, by varying f:

$$\varepsilon_C = \sqrt{\frac{\sum_{\text{bins}}(C^{(k)} - C^{(K)})^2}{N_b \mathrm{E}[C^{(K)}]^2}}, \tag{A.7}$$

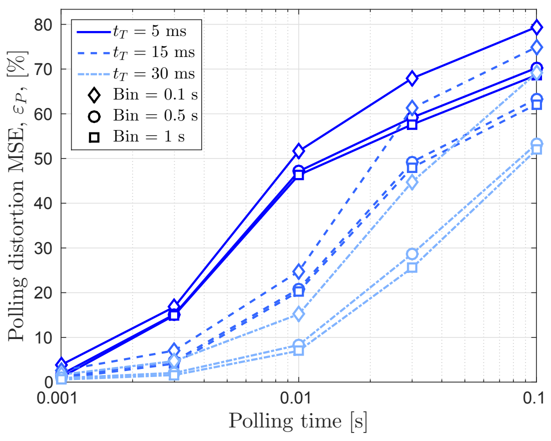
where N_b is the number of bins in a flow. The computations have been repeated for different bin sizes varying in $\{1, 0.5, 0.2, 0.1\}$ seconds (dotted, dash-dotted, dashed and solid lines, respectively). It can be seen that the error decreases below 20 % when more than 20 % of the samples are used.

Figure A.3(b) can also be interpreted as the width of the probability distribution of having an exact measurement using f % of the samples. In particular, it is easy to see that when we use all the samples, the distribution should collapse into a delta function (zero width), while the fewer samples we use, the wider the distribution. The real value can only be larger than the measured one, because of Equation (A.3) that shows $\max_{i\in[1,\dots,k]} c_{W_i}(i) \le \max_{i\in[1,\dots,K]} c_{W_i}(i)$. Thus, this distribution has non-zero width for values smaller than the actual measurement only.

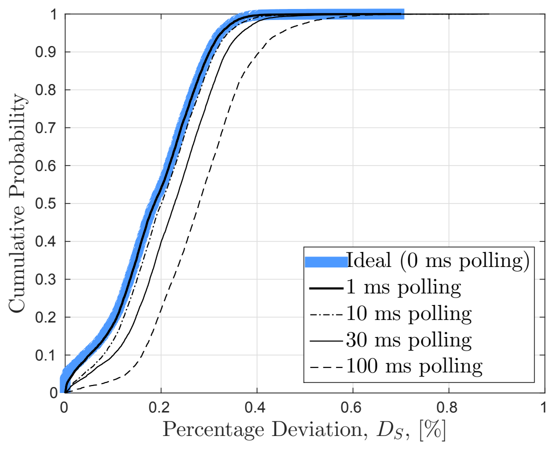
To complete this preliminary evaluation of our measurement technique, Figure A.3(c) shows the variation of the per user capacity ${C_U}^{(K)}(t)$ measured every 500 ms and its estimates ${C_U}^{(k)}(t)$ computed with $f = k/K = \{10, 20, 50, 100\}$ % (dotted, dash-dotted, dashed and solid lines, respectively). Although with 10 % of samples the estimates are quite different from the actual capacity values, we will be showing next that it is possible to exploit these coarse estimates to obtain a sufficiently accurate capacity estimate.

A.3. Simulation Campaign

We have performed an extensive simulation campaign in order to evaluate our proposed technique in a controlled environment. We use a modified version of ns-3.23 [263] and its LTEmodule LENA [264]. We focus on LTEdue to its increasing popularity. In all simulations the monitored user uses TCP, since it is both the most challenging and the most popular [230] transport layer protocol of mobile phones. The variable parameters of the simulations are presented in table A.1. The fixed parameters are: 1) the simulation lasts for 22 seconds and 2) the BS uses a proportionally fair scheduler. For each set of parameters we run the simulation multiple times with a different seed, generating in total 18570 flows.



(a) CV(NRMSE) ε_P



(b) Deviation CDF

Figure A.4: Left: CV(NRMSE) ε_P of the capacity estimate between ideal arrivals ($t_P = 0$) and arrivals that suffer from polling ($t_P \neq 0$), for varying bin sizes and minimum dispersion times t_T. Right: Deviation of the sampling estimations ($k = 5\%$) for various average polling periods t_P from the ideal case ($k = 100\%$, $t_P = 0$).

Table A.1: Simulation parameters

Parameter	Value
number of resource blocks (MHz)	25 (5), 50 (10), 75 (15), 100 (20)
number of competing UEs in the cell	$[0, 1, 2 \dots, 10]$
distance between UE and BS in m	$[0, 50, 100 \dots, 450]$
number of interfering BS	$[0, 1, 2 \dots, 6]$
type of scenario	“static”, “urban walking”, “vehicular”

Next we investigate the effect of polling on the accuracy of the measurements. The simulation results do not suffer from polling, thus the packet arrival time reported in the logs is the actual arrival time at the Networks Interface Card (NIC). In order to simulate the polling effect we manipulate the logs so that we check for incoming packets every $t_P \pm 10\%$, where $t_P \in [1, 3, 10, 30, 100]$ ms. We add the 10% deviation in the timing of each polling because based on our traces and the literature, polling does not have a fixed frequency. We also add a tiny inter-packet delay (in the range of 0.1 ms) between the packets that are reported together by the polling function, in a fashion similar to the one we observe in our traces. Please note that the polling delay (if present) is usually within 10 ms under normal circumstances.

Figure A.4(a) shows the CV(NRMSE) ε_P between traces that have the original timestamps and processed ones. We calculate the ε_P as we did for the ε_C in Equation (A.7).

$$\varepsilon_P = \sqrt{\frac{\sum_{\text{bins}}(C^{(t_P)} - C^{(0)})^2}{N_b \mathrm{E}[C^{(0)}]^2}} \tag{A.8}$$

It can be seen that the error is at most 20% for most cases (up to 10 ms of delay).

Subsequently, we examine how the combination of sampling only 5% of the available estimators and polling affects the accuracy of the results. We divide every flow to 100 ms bins and for every bin we calculate the ${C_U}^{(100\%)}$ and the ${C_U}^{(5\%)}$ for various t_P values. The speed of each flow is the average of the measured capacity of all its bins $\mathrm{E}[{C_U}^{(k)}]$. As a ground truth, against which we compare the rest of the results, we suppose the case where $t_P = 0$ (ideal polling) and $k = K$. Figure A.4(b) depicts the Empirical CDF of the percent Deviation D_S computed by the formula:

$$D_S = \frac{|\mathrm{E}[{C_U}^{(5\%)(t_P)}] - \mathrm{E}[{C_U}^{(100\%)(0)}]|}{\mathrm{E}[{C_U}^{(100\%)(0)}]} \tag{A.9}$$

By comparing the ideal line of $t_P = 0$ with the rest, we conclude that even though polling does have a negative effect in the measurements, the dominant cause of error is the sampling. Also, we observe that for the most common t_P values ($t_P < 10$ ms) the deviation for 90% of the cases is less than 30%.

A.4. Measurement Campaign

In order to validate our measurement technique over many different scenarios and configurations, we organized a measurement campaign that covers two cities in two different countries, Darmstadt (Germany) [17] and Madrid (Spain), for 24 hours a day lasting 7 days. During this time, 5 people per city moved around as they normally do, carrying one measuring device each and performing their usual tasks involving mobile networking on the measuring devices. In order to be able to compare results of both passive and active measurements, we also perform automated periodic file downloads.

All the devices were running a simple Android application, which was periodically sampling the available capacity by starting two download types: *short* downloads of 500 KB to study the TCP slow start phases and *long* downloads of 2 MB to measure TCP steady state throughput. The two types were organized in a sequence with a long download, preceded by two small downloads and later succeeded by another two. We use tcpdump on the measurement devices to monitor the arrival time and size of all incoming packets. The download sequence was repeated every 50 minutes. Additionally, we log other related phone parameters: GPS, cell ID, Channel Quality Indicators (Arbitrary Strength Unit (ASU), dBm) and network technology (2G, 3G, LTE).

The phones used in the campaign were the following: 5 Nexus 5, located in Germany, and 4 Sony Xperia Miro and 1 Samsung Galaxy S3, located in Spain. Also, while the Nexus 5 phones are LTEcapable, the other phones only support radio technologies up to HSPA.

A.5. Results and Discussion

We verified our measurement technique by analyzing more than 3000 unique TCP flows extracted from the communication of the phones participating in the campaign. As before, we split each flow into 100 ms bins and calculate the ${C_U}^{(100\%)}$ and ${C_U}^{(5\%)}$ metrics, and assume that their

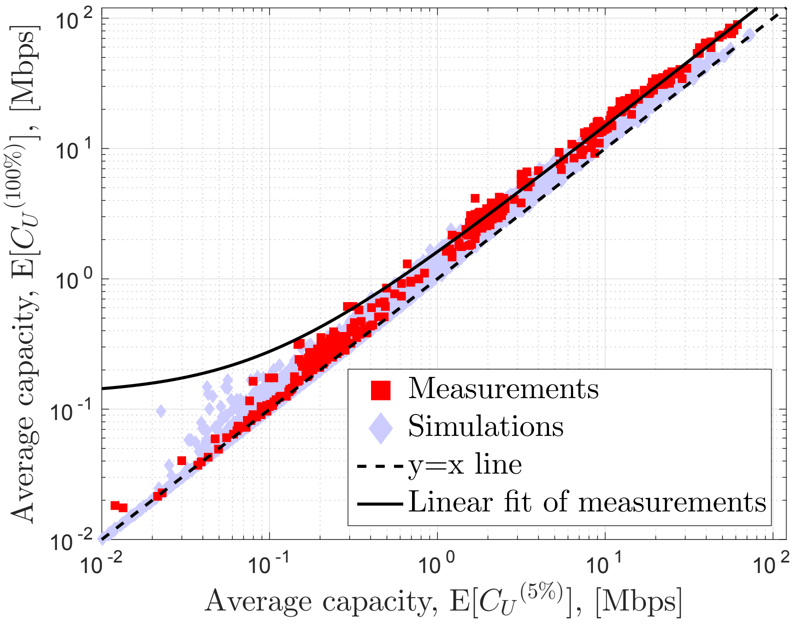


Figure A.5: Scatterplot of the average estimate of per user capacity $\mathrm{E}[\tilde{C}^{(K)}]$ computed over 5 % of the available information ($K = N/20$) against the estimated end-to-end throughput measured using all available information. The dashed line shows the linear regression between the two quantities.

average is the speed of each flow. Note that in these measurements we neither have control over the polling, nor we can distinguish it from the scheduling behavior.

Figure A.5 shows a scatterplot where the abscissa and the ordinate of each rectangular point are the sampled and non-sampled versions of C_U, respectively. Further we add in the same plot the related simulation results for $t_P = 3$ ms as diamonds. As expected from Equation (A.3) all the data points are above the $y = x$ line. Thus, we verify that our algorithm may only underestimate the capacity.

The fact that all the points are so close to the $y = x$ line proves that the values derived by just 5% of the samples are good estimators of $C_U{}^{(100\%)}$. As a consequence, this measurement can be safely used as a lower bound in resource optimization problems. We also plot the linear regression of only the actual measurement results as a dashed line. The regression line would allow us to build an even better estimator with lower error.

The figure is plotted in double logarithmic scale in order to emphasize that the relationship between $C_U{}^{(100\%)}$ and $C_U{}^{(5\%)}$ can be observed over all the measured connection rates and there is an almost constant ratio between the estimate and the actual value. Although outliers are visible, we can obtain quite an accurate estimate of C_U by exploiting as few as 5 % of the packets sent during a TCP connection. This allows for quite an effective passive monitoring technique as, even by monitoring small data exchanges, it is possible to obtain frequent and accurate mobile per user capacity measurements necessary for user throughput prediction and resource allocation. The linear regression line seems to deviate from the measurement for low values of capacity, because of the double logarithmic scale used in the plot, which highlights the regression offset for low values (500 Kbps and less). Further, we observe that for high values, the regression line has an almost fixed vertical distance from the $y = x$ line (constant percentage error). This represents the error of the estimate and, since it is constant, in the double logarithmic plot, appears as a fixed deviation on the Y-axis from the $y = x$ line.

Unfortunately, using very low rate background traffic is impossible. The rates of such traffic are on the order of 4 packets over 100 ms, which do not allow for reliable capacity measurements. Also, a big number of the APPs use the Google Cloud Messaging (GCM) service, which minimizes their notification related traffic. In the case of GCM, if there is an update a few packets are sent just to generate a notification. When the user interacts with the notification, a larger number of packets are downloaded. In this scenario, we can use that download to get an estimation.

In the experiments, we use rooted Android phones and tcpdump to perform the measurements. Given the very low complexity and resources that are required by our approach, the C_U estimation is generated at virtually no cost. Therefore, we believe that it may be included in the OS as a service to applications that may opt-in to use it. For example, the flow-id, the timestamp and the size of a packet could be registered as part of the standard kernel packet processing procedure. Since these values do not contain any sensitive information, there are no privacy concerns and after a short period to time, when this information is irrelevant it can be deleted. Upon application request, the OS could generate a C_U estimation, if there are sufficient data stored. The knowledge of the flow-id can help distinguish the state of a TCP flow (slow-start, steady-state etc.). If it is possible to use small values of t_T, it is possible to generate accurate estimators even during the late part of slow start, when the congestion/receive windows have relatively high values, since then the dispersion time can be smaller than the time required by the antenna to transmit a server burst. In case of a TCP flow that stops very early, it can be difficult to remove both the slow start and the scheduling artifacts. In such cases, the resulting value will be significantly lower than the truth, but this is easy to detect and filter (e.g., requiring a flow to generate at least 75 downlink packets in order to be used).

As a side note, our technique is also able to estimate fast per user capacity variations. However, it obtains a lower accuracy since a larger fraction of samples are needed to estimate the maximum of the c_W distribution. Nonetheless, it is often sufficient to use 20 $\%$ of the samples collected in a bin to achieve a reasonable estimate of C_U. In fact, with the smallest bin size and as few as 20 $\%$ of the samples have an error $\varepsilon_C < 0.2$, which means the actual capacity should not be larger than 120 $\%$ of the estimated value.

In addition, t_T must be taken slightly longer than the TTI to avoid the measurement being impacted by many bursty arrivals. In line with Equation (A.1) of Section A.2, $\Delta(t_T)$ approaches zero for $t_T > 15$ ms for most of the recorded flows.

Figure A.6 shows the CV(NRMSE) for various combinations of t_T and f of the measurement campaign flows. The bin size is set to 200 ms to give an example of this technique's results when it collects very frequent measurements. As expected ε_C decreases when t_T and f increase. For

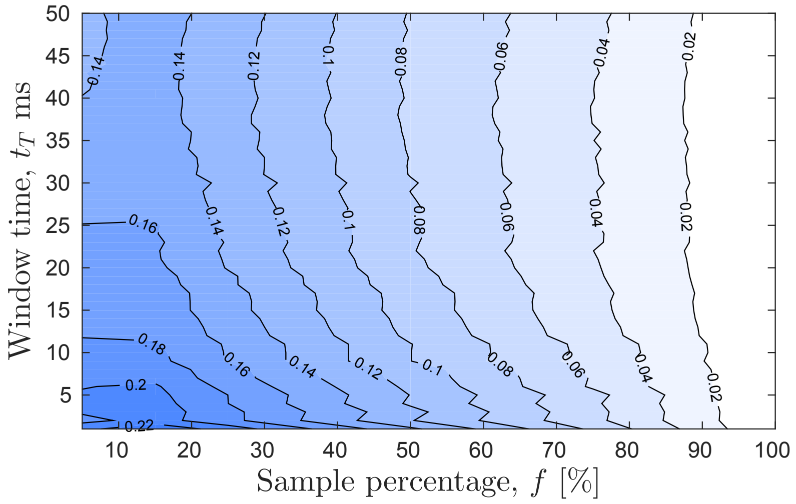


Figure A.6: Contour graph of ε_C varying t_T and f for a bin size of 200 ms.

values of $t_T \geq 15$ ms and $f \geq 20$ %, the error is small enough for the model to give trustworthy results ($\varepsilon_C \leq 15$ %).

Finally, Table A.2 shows some of the overall evaluation of the traces obtained by the measurement campaign with $f = 25$ % averaged over the bin size and using the optimal t_T ($\min t_T|\Delta(t_T) \rightarrow 0$). Optimal t_T and C_U are computed as described in Section A.2 and then averaged over all the traces. While some of the flows are transmitted using 2G EDGE data, the results are not included since there are too few such flows for statistical significance.

Technology	UMTS	HSPA	HSPA+	LTE
C_U (Mbps)	10.83	1.4	10.74	24.3
Optimal t_T (ms)	19	23	17	16

Table A.2: Average C_U and average optimal t_T per technology.

The measurements are based on the data reported by the Android OS. Note that HSPA and HSPA+ are a family of enhancements to UMTS, that greatly increase its speed. The high average speed of UMTS is related to networks that support the High Speed Downlink Packet Access (HSDPA) enhancement for improved downlink speed, but not all the enhancements that would classify them as HSPA or HSPA+. The very big differences in speed between the HSPA, HSPA+ and LTEtechnologies can be explained by the following reasons. More recent technologies can achieve higher speeds. Smartphones tend to use the best technology possible for their channel quality. Thus, they use HSPA only when their signal is too bad to use a better technology and in turn the bad signal greatly affects speed.

Our approach is designed for downlink measurements, which account for the vast majority of the smartphone generated traffic [230]. Recent trends, though, show an increase in uplink related user activity and therefore we will briefly discuss the uplink case. Our algorithm cannot be directly applied to the uplink due to uplink communication characteristics. For instance, if we attempt to perform a measurement on the phone side we can gather very limited information. Without accessing the transceiver firmware, we can only observe how fast packets appear in the kernel, instead of how fast the NIC successfully transmits them at the medium, which is the metric we are interested in. It is possible that packets may remain in the buffer of the NIC for a relatively long time after they appear in the kernel, leading to wrong estimations. On the other hand, applying our algorithm to measurements collected on the server side will fail to measure the cell capacity, since many intermediate hops may be between the eNodeB and the server. An alternative approach would be to infer clues of the speed indirectly at the phone side. If a UDP socket is blocking, it can be an indication that the rate at which an application is generating packets (which we can detect) is higher than the link capacity, thus deriving an upper limit of the speed. In the case of TCP traffic, the ACKs can be analyzed to infer whether the rate that the application is generating traffic is above or below the link capacity. Further analyzing the uplink scenario is beyond the scope of the present paper and we leave it for future work.

References

[1] T. Chiang, "What's expected of us," *Nature*, vol. 436, no. 7047, pp. 150–150, 2005.

[2] Cisco VNI, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021," *Cisco Public Information*, 2016.

[3] N. Bui, M. Cesana, A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A survey of anticipatory mobile networking: Context-based classification, prediction methodologies, and optimization techniques," *Submitted to IEEE Communications Surveys and Tutorials*, 2017.

[4] N. Bui, F. Michelinakis, and J. Widmer, "A Model for Throughput Prediction for Mobile Users," in *European Wireless 2014*, 2014, pp. 1–6.

[5] N. Bui and J. Widmer, "Modelling throughput prediction errors as gaussian random walks," in *The 1st KuVS Workshop on Anticipatory Networks*, 2014.

[6] ——, "Mobile network resource optimization under imperfect prediction," in *IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2015, pp. 1–9.

[7] N. Bui, S. Valentin, and J. Widmer, "Anticipatory quality-resource allocation for multi-user mobile video streaming," in *IEEE Communication and Networking Techniques for Contemporary Video (CNTCV - INFOCOM workshop)*, 2015.

[8] N. Bui, I. Malanchini, and J. Widmer, "Anticipatory admission control and resource allocation for media streaming in mobile networks," in *ACM Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, 2015.

[9] N. Bui and J. Widmer, "Data-driven Evaluation of Anticipatory Networking Optimization on LTE Networks," in *submitted to IEEE ITC29*, 2017.

[10] ——, "Data-driven Evaluation of Anticipatory Networking Optimization on LTE Networks," in *submitted to IEEE Transactions on Mobile Computing*, 2017.

[11] ——, "OWL: a Reliable Online Watcher for LTE Control Channel Measurements," in *ACM All Things Cellular (MobiCOM workshop)*, Oct. 2016.

[12] F. Michelinakis, N. Bui, G. Fioravantti, J. Widmer, F. Kaup, and D. Hausheer, “Lightweight mobile bandwidth availability measurement,” in *IFIP Networking Conference*, 2015, pp. 1–9.

[13] F. Michelinakis, N. Bui, G. Fioravantti, F. Kaup, D. Hausheer, and J. Widmer, “Lightweight mobile bandwidth availability measurement,” *Elsevier Computer Communications*, vol. 84, pp. 73–83, Jun. 2016.

[14] C. Koch, J. Ruckert, N. Bui, F. Michelinakis, G. Fioravantti, D. Hausheer, and J. Widmer, “Demo: Mobile social prefetcher using social and network information,” in *IEEE International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*, 2014.

[15] C. Koch, N. Bui, J. Rückert, G. Fioravantti, F. Michelinakis, S. Wilk, J. Widmer, and D. Hausheer, “Media download optimization through prefetching and resource allocation in mobile networks,” in *Proceedings of the 6th ACM Multimedia Systems Conference (MMSys)*, 2015, pp. 85–88.

[16] F. Kaup, F. Michelinakis, N. Bui, J. Widmer, K. Wac, and D. Hausheer, “Behind the NAT–a measurement based evaluation of cellular service quality,” in *IEEE Conference on Network and Service Management (CNSM)*, 2015, pp. 228–236.

[17] ——, “Assessing the implications of cellular network performance on mobile content access,” *IEEE Transactions on Network and Service Management*, vol. 12, pp. 168–180, Mar. 2016.

[18] P. Dini, M. Miozzo, N. Bui, and N. Baldo, “A model to analyze the energy savings of base station sleep mode in LTE HetNets,” in *International Conference on Green Computing and Communications (GreenCom)*, 2013, pp. 1375–1380.

[19] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, “Internet of Things for smart cities,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[20] N. Bui and M. Rossi, “Staying alive: System design for self-sufficient sensor networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 11, no. 3, p. 40, 2015.

[21] Z. Lu and G. De Veciana, “Optimizing stored video delivery for mobile networks: The value of knowing the future,” in *IEEE INFOCOM*, 2013, pp. 2706–2714.

[22] H. Abou-zeid, H. S. Hassanein, and S. Valentin, “Optimal predictive resource allocation: Exploiting mobility patterns and radio maps,” in *IEEE Global Communications Conference (GLOBECOM)*, 2013, pp. 4877–4882.

[23] V. A. Siris and D. Kalyvas, “Enhancing mobile data offloading with mobility prediction and prefetching,” *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 17, no. 1, pp. 22–29, 2013.

[24] H. Abou-zeid, H. S. Hassanein, Z. Tanveer, and N. AbuAli, “Evaluating mobile signal and location predictability along public transportation routes,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2015, pp. 1195–1200.

[25] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, and A. T. Campbell, “Nextplace: a spatio-temporal prediction framework for pervasive systems,” in *Springer Pervasive Computing*, 2011, vol. 6696, pp. 152–169.

[26] M. De Domenico, A. Lima, and M. Musolesi, “Interdependence and predictability of human mobility and social interactions,” *Elsevier Pervasive and Mobile Computing*, vol. 9, no. 6, pp. 798–807, 2013.

[27] Y. Jiang, D. C. Dhanapala, and A. P. Jayasumana, “Tracking and prediction of mobility without physical distance measurements in sensor networks,” in *IEEE International Conference on Communications (ICC)*, 2013, pp. 1845–1850.

[28] Q. Liao, S. Valentin, and S. Stanczak, “Channel gain prediction in wireless networks based on spatial-temporal correlation,” in *IEEE Signal Processing Advances in Wireless Communications (SPAWC)*, 2015, pp. 400–404.

[29] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, “Video streaming using a location-based bandwidth-lookup service for bitrate planning,” *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 8, no. 3, pp. 24:1–24:19, 2012.

[30] J. Yang and Z. Fei, “Broadcasting with prediction and selective forwarding in vehicular networks,” *Hindawi International journal of distributed sensor networks*, vol. 2013.

[31] Y. Chon, E. Talipov, H. Shin, and H. Cha, “SmartDC: Mobility prediction-based adaptive duty cycling for everyday location monitoring,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 3, pp. 512–525, 2014.

[32] ——, “Mobility prediction-based smartphone energy optimization for everyday location monitoring,” in *ACM conference on embedded networked sensor systems (SenSys)*, 2011, pp. 82–95.

[33] X. Chen, F. Mériaux, and S. Valentin, “Predicting a user’s next cell with supervised learning based on channel states,” in *IEEE Signal Processing Advances in Wireless Communications (SPAWC)*, 2013, pp. 36–40.

[34] Y. Chon, N. D. Lane, Y. Kim, F. Zhao, and H. Cha, "Understanding the coverage and scalability of place-centric crowdsensing," in *ACM international joint conference on Pervasive and ubiquitous computing (Ubicomp)*, 2013, pp. 3–12.

[35] J. Froehlich and J. Krumm, "Route prediction from trip observations," SAE Technical Paper, Tech. Rep., 2008.

[36] A. Monreale, F. Pinelli, R. Trasarti, and F. Giannotti, "WhereNext: a location predictor on trajectory pattern mining," in *ACM international conference on Knowledge discovery and data mining (SIGKDD)*, 2009, pp. 637–646.

[37] L. Ghouti, T. R. Sheltami, and K. S. Alutaibi, "Mobility prediction in mobile ad hoc networks using extreme learning machines," *Procedia Computer Science*, vol. 19, pp. 305–312, 2013.

[38] J. Hao, R. Zimmermann, and H. Ma, "Gtube: Geo-predictive video streaming over http in mobile environments," in *ACM Multimedia Systems Conference (MMSys)*, 2014, pp. 259–270.

[39] R. Margolies, A. Sridharan, V. Aggarwal, R. Jana, N. Shankaranarayanan, V. Vaishampayan, G. Zussman *et al.*, "Exploiting mobility in proportional fair cellular scheduling: Measurements and algorithms," in *IEEE INFOCOM*, 2014, pp. 1339–1347.

[40] A. Sridharan and J. Bolot, "Location patterns of mobile users: A large-scale study," in *IEEE INFOCOM*, 2013, pp. 1007–1015.

[41] N. Namvar, W. Saad, B. Maham, and S. Valentin, "A context-aware matching game for user association in wireless small cell networks," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 439–443.

[42] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási, "Limits of predictability in human mobility," *Science*, vol. 327, no. 5968, pp. 1018–1021, 2010.

[43] J.-K. Lee and J. C. Hou, "Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application," in *ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2006, pp. 85–96.

[44] H. Abu-Ghazaleh and A. S. Alfa, "Application of mobility prediction in wireless networks using Markov renewal theory," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 2, pp. 788–802, 2010.

[45] D. Barth, S. Bellahsene, and L. Kloul, "Mobility prediction using mobile user profiles," in *IEEE Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2011, pp. 286–294.

[46] ——, “Combining local and global profiles for mobility prediction in LTE femtocells,” in *ACM Modeling, analysis and simulation of wireless and mobile systems (MSWiM)*, 2012, pp. 333–342.

[47] Y. Chon, H. Shin, E. Talipov, and H. Cha, “Evaluating mobility models for temporal prediction with high-granularity mobility data,” in *IEEE Pervasive Computing and Communications (PerCom)*, 2012, pp. 206–212.

[48] G. Gidófalvi and F. Dong, “When and where next: Individual mobility prediction,” in *ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, 2012, pp. 57–64.

[49] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson, “Approaching the limit of predictability in human mobility,” *Nature Scientific reports*, vol. 3, 2013.

[50] H. Xiong, D. Zhang, D. Zhang, V. Gauthier, K. Yang, and M. Becker, “MPaaS: Mobility prediction as a service in telecom cloud,” *Springer Information Systems Frontiers*, vol. 16, no. 1, pp. 59–75, 2014.

[51] Y. Chon, Y. Kim, H. Shin, and H. Cha, “Adaptive duty cycling for place-centric mobility monitoring using zero-cost information in smartphone,” *IEEE Transactions on Mobile Computing*, vol. 13, no. 8, pp. 1694–1706, 2014.

[52] H. Abou-zeid, H. S. Hassanein, and S. Valentin, “Energy-efficient adaptive video transmission: Exploiting rate predictions in wireless networks,” *IEEE Transactions on Vehicular Technology*, vol. 63, no. 5, pp. 2013–2026, 2014.

[53] H. Abou-zeid and H. S. Hassanein, “Efficient lookahead resource allocation for stored video delivery in multi-cell networks,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2014, pp. 1909–1914.

[54] M. Dräxler and H. Karl, “Cross-layer scheduling for multi-quality video streaming in cellular wireless networks,” in *IEEE International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2013, pp. 1181–1186.

[55] M. Dräxler, J. Blobel, P. Dreimann, S. Valentin, and H. Karl, “SmarterPhones: Anticipatory download scheduling for wireless video streaming,” in *IEEE International Conference and Workshops on Networked Systems (NetSys)*, 2015, pp. 1–8.

[56] S. Valentin, “Anticipatory resource allocation for wireless video streaming,” in *IEEE International Conference on Communication Systems (ICCS)*, 2014, pp. 107–111.

[57] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, “Can accurate predictions improve video streaming in cellular networks?” in

ACM International Workshop on Mobile Computing Systems and Applications (HotMobile), 2015, pp. 57–62.

[58] L. S. Muppirisetty, J. Tadrous, A. Eryilmaz, and H. Wymeersch, "On proactive caching with demand and channel uncertainties," in *IEEE Conference on Communication, Control, and Computing (Allerton)*, 2015, pp. 1174–1181.

[59] M. Fr, L. S. Muppirisetty, H. Wymeersch *et al.*, "Channel gain prediction for multi-agent networks in the presence of location uncertainty," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, pp. 3911–3915.

[60] M. Dräxler, J. Blobel, and H. Karl, "Anticipatory download scheduling in wireless video streaming with uncertain data rate prediction," in *IFIP Wireless and Mobile Networking Conference (WMNC)*, 2015, pp. 136–143.

[61] D. Tsilimantos, A. Nogales-Gómez, and S. Valentin, "Anticipatory Radio Resource Management for Mobile Video Streaming with Linear Programming," in *IEEE International Conference on Communications (ICC)*, 2016.

[62] R. Atawia, H. Abou-zeid, H. S. Hassanein, and A. Noureldin, "Robust resource allocation for predictive video streaming under channel uncertainty," in *IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 4683–4688.

[63] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi, "Video through a crystal ball: effect of bandwidth prediction quality on adaptive streaming in mobile environments," in *ACM International Workshop on Mobile Video (MoVid)*, 2016, p. 1.

[64] R. Atawia, H. Abou-zeid, H. S. Hassanein, and A. Noureldin, "Chance-constrained qos satisfaction for predictive video streaming," in *IEEE Local Computer Networks (LCN)*, 2015, pp. 253–260.

[65] ——, "Joint chance-constrained predictive resource allocation for energy-efficient video streaming," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 34, no. 5, pp. 1389–1404, 2016.

[66] Z. Liu and Y. Wei, "Hop-by-hop adaptive video streaming in content centric network," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[67] S. Naimi, A. Busson, V. Vèque, L. B. H. Slama, and R. Bouallegue, "Anticipation of ETX metric to manage mobility in ad hoc wireless networks," in *Springer Ad-hoc, Mobile, and Wireless Networks*, 2014, pp. 29–42.

[68] X. Wang, M. Chen, T. T. Kwon, L. Yang, and V. Leung, "AMES-Cloud: a framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Transactions on Multimedia*, vol. 15, no. 4, pp. 811–820, 2013.

[69] E. Kurdoglu, Y. Liu, Y. Wang, Y. Shi, C. Gu, and J. Lyu, “Real-time bandwidth prediction and rate adaptation for video calls over cellular networks,” in *ACM International Conference on Multimedia Systems (MMSys)*, 2016, p. 12.

[70] W. Bao and S. Valentin, “Bitrate adaptation for mobile video streaming based on buffer and channel state,” in *IEEE International Conference on Communications (ICC)*, 2015, pp. 3076–3081.

[71] D. Bianchi, A. Ferrara, and M. Di Benedetto, “Networked model predictive traffic control with time varying optimization horizon: The Grenoble South Ring case study,” in *IEEE European Control Conference (ECC)*, 2013, pp. 4039–4044.

[72] S. Yin, D. Chen, Q. Zhang, and S. Li, “Prediction-based throughput optimization for dynamic spectrum access,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 1284–1289, 2011.

[73] S. A. Hosseini, F. Fund, and S. S. Panwar, “(Not) yet another policy for scalable video delivery to mobile users,” in *ACM International Workshop on Mobile Video (MoVid)*, 2015, pp. 17–22.

[74] E. Dall’Anese, S.-J. Kim, and G. B. Giannakis, “Channel gain map tracking via distributed Kriging,” *IEEE Transactions on Vehicular Technology*, vol. 60, no. 3, pp. 1205–1211, 2011.

[75] M. Kasparick, R. L. Cavalcante, S. Valentin, S. Stanczak, and M. Yukawa, “Kernel-based adaptive online reconstruction of coverage maps with side information,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 7, pp. 5461–5473, 2015.

[76] L. S. Muppirisetty, T. Svensson, and H. Wymeersch, “Spatial wireless channel prediction under location uncertainty,” *IEEE Transactions on Wireless Communications*, vol. 15, no. 2, pp. 1031–1044, 2016.

[77] M. Piacentini and F. Rinaldi, “Path loss prediction in urban environment using learning machines and dimensionality reduction techniques,” *Springer Computational Management Science*, vol. 8, no. 4, pp. 371–385, 2011.

[78] S. J. Tarsa, M. Comiter, M. B. Crouse, B. McDanel, and H. Kung, “Taming Wireless Fluctuations by Predictive Queuing Using a Sparse-Coding Link-State Model,” in *ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2015, pp. 287–296.

[79] X. Tie, A. Seetharam, A. Venkataramani, D. Ganesan, and D. L. Goeckel, “Anticipatory wireless bitrate control for blocks,” in *ACM COnference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2011, p. 9.

[80] O. Semiari, W. Saad, S. Valentin, M. Bennis, and H. V. Poor, "Context-aware small cell networks: How social metrics improve wireless resource allocation," *IEEE Transactions on Wireless Communications*, vol. 14, no. 11, pp. 5927–5940, 2015.

[81] A. J. Nicholson and B. D. Noble, "Breadcrumbs: forecasting mobile connectivity," in *ACM international conference on Mobile computing and networking (MobiCom)*, 2008, pp. 46–57.

[82] A. Seetharam, P. Dutta, V. Arya, J. Kurose, M. Chetlur, and S. Kalyanaraman, "On managing quality of experience of multiple video streams in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 3, pp. 619–631, 2015.

[83] P. Fazio, M. Tropea, F. De Rango, and M. Voznak, "Pattern prediction and passive bandwidth management for hand-over optimization in QoS cellular networks with vehicular mobility," *IEEE Transactions on Mobile Computing*.

[84] N. Abedini and S. Shakkottai, "Content caching and scheduling in wireless networks with elastic and inelastic traffic," *IEEE/ACM Transactions on Networking*, vol. 22, no. 3, pp. 864–874, 2014.

[85] H. Abou-Zeid and H. S. Hassanein, "Predictive green wireless access: Exploiting mobility and application information," *IEEE Wireless Communications*, vol. 20, no. 5, pp. 92–99, 2013.

[86] L. Huang, S. Zhang, M. Chen, and X. Liu, "When backpressure meets predictive scheduling," in *ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, 2014, pp. 33–42.

[87] M. Proebster, M. Kaschub, and S. Valentin, "Context-aware resource allocation to improve the quality of service of heterogeneous traffic," in *IEEE International Conference on Communications (ICC)*, 2011, pp. 1–6.

[88] J. Tadrous, A. Eryilmaz, and H. El Gamal, "Proactive resource allocation: Harnessing the diversity and multicast gains," *IEEE Transactions on Information Theory*, vol. 59, no. 8, pp. 4833–4854, 2013.

[89] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in high-speed mobility using bandwidth maps," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 603–617, 2012.

[90] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A control-theoretic approach for dynamic adaptive video streaming over HTTP," *ACM SIGCOMM Computer Communication Review*, vol. 45, no. 4, pp. 325–338, 2015.

[91] A. H. Zahran, J. Quinlan, D. Raca, C. J. Sreenan, E. Halepovic, R. K. Sinha, R. Jana, and V. Gopalakrishnan, "OSCAR: an optimized stall-cautious adaptive bitrate streaming

algorithm for mobile networks," in *ACM International Workshop on Mobile Video (MoVid)*, 2016, p. 2.

[92] E. Pollakis and S. Stanczak, "Anticipatory networking for energy savings in 5G systems," *VDE ITG-Fachbericht-WSA*, 2016.

[93] C. Wang, A. Rizk, and M. Zink, "Squad: a spectrum-based quality adaptation for dynamic adaptive streaming over http," in *ACM International Conference on Multimedia Systems (MMSys)*, 2016, p. 1.

[94] H. Yu, M. H. Cheung, L. Huang, and J. Huang, "Power-delay tradeoff with predictive scheduling in integrated cellular and wi-fi networks," *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 34, no. 4, pp. 735–742, 2016.

[95] ——, "Predictive delay-aware network selection in data offloading," in *IEEE Global Communications Conference (GLOBECOM)*, 2014, pp. 1376–1381.

[96] J. Du, C. Jiang, Y. Qian, Z. Han, and Y. Ren, "Traffic prediction based resource configuration in space-based systems," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[97] ——, "Resource allocation with video traffic prediction in cloud-based space systems," *IEEE Transactions on Multimedia*, vol. 18, no. 5, pp. 820–830, 2016.

[98] K. Miller, D. Bethanabhotla, G. Caire, and A. Wolisz, "A control-theoretic approach to adaptive video streaming in dense wireless networks," *IEEE Transactions on Multimedia*, vol. 17, no. 8, pp. 1309–1322, 2015.

[99] M.-F. R. Lee, F.-H. S. Chiu, H.-C. Huang, and C. Ivancsits, "Generalized predictive control in a wireless networked control system," *Hindawi International Journal of Distributed Sensor Networks*, 2013.

[100] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory," *Elsevier Transportation Research Part B: Methodological*, vol. 18, no. 1, pp. 1–11, 1984.

[101] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of internet backbone traffic: Observations and initial models," in *IEEE INFOCOM*, 2003, pp. 1178–1188.

[102] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute path bandwidth traces from 3G networks: Analysis and applications," in *ACM Multimedia Systems Conference (MMSys)*, 2013, pp. 114–118.

[103] N. Sadek and A. Khotanzad, “Multi-scale high-speed network traffic prediction using k-factor Gegenbauer ARMA model,” in *IEEE International Conference on Communications (ICC)*, vol. 4, 2004, pp. 2148–2152.

[104] B. Zhou, D. He, Z. Sun, and W. H. Ng, “Network traffic modeling and prediction with ARIMA/GARCH,” in *HET-NETs Conference*, 2005, pp. 1–10.

[105] F. Fu and M. van der Schaar, “A systematic framework for dynamically optimizing multi-user wireless video transmission,” *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 308–320, 2010.

[106] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, “CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction,” in *ACM SIGCOMM*, 2016, pp. 272–285.

[107] C. Chen, X. Zhu, G. de Veciana, A. C. Bovik, and R. W. Heath, “Rate adaptation and admission control for video transmission with subjective quality constraints,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 1, pp. 22–36, 2015.

[108] S. Samulevicius, T. B. Pedersen, and T. B. Sorensen, “MOST: mobile broadband network optimization using planned spatio-temporal events,” in *IEEE Vehicular Technology Conference (VTC Spring)*, 2015, pp. 1–5.

[109] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das, “Understanding traffic dynamics in cellular data networks,” in *IEEE INFOCOM*, 2011, pp. 882–890.

[110] Z. Sayeed, Q. Liao, D. Faucher, E. Grinshpun, and S. Sharma, “Cloud analytics for wireless metric prediction-framework and performance,” in *IEEE International Conference on Cloud Computing (CLOUD)*, 2015, pp. 995–998.

[111] A. B. V. Sekar, A. Akella, S. S. I. Stoica, and H. Zhang, “Developing a predictive model of quality of experience for internet video,” in *ACM SIGCOMM*, 2013, pp. 339–350.

[112] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang, “Characterizing and modeling internet traffic dynamics of cellular devices,” in *ACM joint international conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2011, pp. 305–316.

[113] Q. Xu, S. Mehrotra, Z. Mao, and J. Li, “PROTEUS: Network Performance Forecast for Real-time, Interactive Mobile Applications,” in *ACM international conference on Mobile systems, applications, and services (MobiSys)*, 2013, pp. 347–360.

[114] J. Jiang, V. Sekar, H. Milner, D. Shepherd, I. Stoica, and H. Zhang, “Cfa: a practical prediction system for video qoe optimization,” in *USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, 2016, pp. 137–150.

[115] P. Millan, C. Molina, E. Dimogerontakis, L. Navarro, R. Meseguer, B. Braem, and C. Blondia, “Tracking and predicting end-to-end quality in wireless community networks,” in *IEEE International Conference on Future Internet of Things and Cloud (FiCloud)*, 2015, pp. 794–799.

[116] O. Semiari, W. Saad, and M. Bennis, “Context-aware scheduling of joint millimeter wave and microwave resources for dual-mode base stations,” in *IEEE International Conference on Communications (ICC)*, 2016.

[117] F. Beister and H. Karl, “Predicting mobile video inter-download times with hidden Markov models,” in *IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, 2014, pp. 359–364.

[118] E. Baştuğ, J.-L. Guénégo, and M. Debbah, “Proactive small cell networks,” in *IEEE International Conference on Telecommunications (ICT)*, 2013.

[119] M. Proebster, M. Kaschub, T. Werthmann, and S. Valentin, “Context-aware resource allocation for cellular wireless networks,” *EURASIP Journal on Wireless Communications and Networking*, vol. 2012, p. 2012:216.

[120] G. Tsiropoulos, D. G. Stratogiannis, N. Mantas, and M. Louta, “The impact of social distance on utility based resource allocation in next generation networks,” in *IEEE International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, 2011.

[121] V. A. Siris, X. Vasilakos, and D. Dimopoulos, “Exploiting mobility prediction for mobility & popularity caching and dash adaptation,” in *IEEE World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2016, pp. 1–8.

[122] K. Witheephanich, J. M. Escaño, D. Muñoz de la Peña, and M. J. Hayes, “A min–max model predictive control approach to robust power management in ambulatory wireless sensor networks,” *IEEE Systems Journal*, vol. 8, no. 4, pp. 1060–1073, 2014.

[123] J. Tadrous and A. Eryilmaz, “On optimal proactive caching for mobile networks with demand uncertainties,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2715–2727, 2015.

[124] J. Tadrous, A. Eryilmaz, and H. El Gamal, “Joint smart pricing and proactive content caching for mobile services,” *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2357–2371, 2015.

[125] W. Wanalertlak, B. Lee, C. Yu, M. Kim, S.-M. Park, and W.-T. Kim, “Behavior-based mobility prediction for seamless handoffs in mobile wireless networks,” *Springer Wireless Networks*, vol. 17, no. 3, pp. 645–658, 2011.

[126] C. Chen, R. W. Heath, A. C. Bovik, and G. de Veciana, “A Markov decision model for adaptive scheduling of stored scalable videos,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 23, no. 6, pp. 1081–1095, 2013.

[127] E. Baştuğ, M. Bennis, and M. Debbah, “Think before reacting: Proactive caching in 5G small cell networks,” *Wiley, submitted*, 2015.

[128] ——, “Living on the edge: The role of proactive caching in 5G wireless networks,” *IEEE Communications Magazine*, vol. 52, no. 8, pp. 82–89, 2014.

[129] ——, “Anticipatory caching in small cell networks: A transfer learning approach,” in *1st KuVS Workshop on Anticipatory Networks*, 2014.

[130] S. Dutta, A. Narang, S. Bhattacherjee, A. S. Das, and D. Krishnaswamy, “Predictive caching framework for mobile wireless networks,” in *IEEE International Conference on Mobile Data Management (MDM)*, 2015, pp. 179–184.

[131] M. C. Mozer, R. Wolniewicz, D. B. Grimes, E. Johnson, and H. Kaushansky, “Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry,” *IEEE Transactions on Neural Networks*, vol. 11, no. 3, pp. 690–696, 2000.

[132] A. Noulas, S. Scellato, N. Lathia, and C. Mascolo, “Mining user mobility features for next place prediction in location-based services,” in *IEEE International Conference on Data Mining (ICDM)*, 2012, pp. 1038–1043.

[133] Z. Yi, X. Dong, X. Zhang, and W. Wang, “Spatial traffic prediction for wireless cellular system based on base stations social network,” in *IEEE Systems Conference (SysCon)*, 2016, pp. 1–5.

[134] K. Hamidouche, W. Saad, and M. Debbah, “Many-to-many matching games for proactive social-caching in wireless small cell networks,” in *IEEE International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2014, pp. 569–574.

[135] Y. Gu, W. Saad, M. Bennis, M. Debbah, and Z. Han, “Matching theory for future wireless networks: fundamentals and applications,” *IEEE Communications Magazine*, vol. 53, no. 5, pp. 52–59, 2015.

[136] H. Bapierre, G. Groh, and S. Theiner, “A variable order Markov model approach for mobility prediction,” *Pervasive Computing*, pp. 8–16, 2011.

[137] F. Calabrese, G. D. Lorenzo, and C. Ratti, “Human mobility prediction based on individual and collective geographical preferences,” in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2010, pp. 312–317.

[138] N. Golrezaei, K. Shanmugam, A. G. Dimakis, A. F. Molisch, and G. Caire, “Femtocaching: Wireless video content delivery through distributed caching helpers,” in *IEEE INFOCOM*, 2012, pp. 1107–1115.

[139] Y. Zhang, E. Pan, L. Song, W. Saad, Z. Dawy, and Z. Han, “Social network aware device-to-device communication in wireless networks,” *IEEE Transactions on Wireless Communications*, vol. 14, no. 1, pp. 177–190, 2015.

[140] K. Zheng, Z. Yang, K. Zhang, P. Chatzimisios, K. Yang, and W. Xiang, “Big data-driven optimization for mobile networks toward 5G,” *IEEE Network*, vol. 30, no. 1, pp. 44–51, 2016.

[141] P. Makris, D. N. Skoutas, and C. Skianis, “A survey on context-aware mobile and wireless networking: On networking and computing environments’ integration,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 362–386, 2013.

[142] V. Pejovic and M. Musolesi, “Anticipatory mobile computing: A survey of the state of the art and research challenges,” *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 47, 2015.

[143] S. Boucheron, O. Bousquet, and G. Lugosi, “Theory of classification: A survey of some recent advances,” *ESAIM: probability and statistics*, vol. 9, pp. 323–375, 2005.

[144] Y. Liu and J. Y. Lee, “An empirical study of throughput prediction in mobile data networks,” in *IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[145] T. T. Nguyen and G. Armitage, “A survey of techniques for internet traffic classification using machine learning,” *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[146] L. Jin, Y. Chen, T. Wang, P. Hui, and A. V. Vasilakos, “Understanding user behavior in online social networks: A survey,” *IEEE Communications Magazine*, vol. 51, no. 9, pp. 144–150, 2013.

[147] S. Baraković and L. Skorin-Kapov, “Survey and challenges of QoE management issues in wireless networks,” *Hindawi Journal of Computer Networks and Communications*, 2013.

[148] M. O. Jackson, *Social and economic networks*. Princeton, NJ, USA: Princeton University Press, 2008.

[149] A. C. Harvey, *Forecasting, structural time series models and the Kalman filter*. Cambridge university press, 1990.

[150] J. Lee, M. Sun, and G. Lebanon, “A comparative study of collaborative filtering algorithms,” *arXiv preprint arXiv:1205.3193*, 2012.

[151] R. Xu, D. Wunsch *et al.*, “Survey of clustering algorithms,” *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[152] S. K. Murthy, “Automatic construction of decision trees from data: A multi-disciplinary survey,” *Kluwer Data mining and knowledge discovery*, vol. 2, no. 4, pp. 345–389, 1998.

[153] J. O. Ramsay and C. Dalzell, “Some tools for functional data analysis,” *JSTOR Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 539–572, 1991.

[154] J. O. Ramsay, *Functional data analysis*. Wiley Online Library, 2006.

[155] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[156] A. Schrijver, *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[157] S. J. Qin and T. A. Badgwell, “A survey of industrial model predictive control technology,” *Elsevier Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

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

[159] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press Cambridge, 1998, vol. 1, no. 1.

[160] “MOMENTUM, MOdels and siMulations for nEtwork plaNning and conTrol of UMts,” 2004. [Online]. Available: http://www.zib.de/momentum

[161] “GeoPKDD: Geographic Privacy-aware Knowledge Discovery and Delivery,” 2005-2008. [Online]. Available: http://www.geopkdd.eu

[162] Telecom Italia, “Big data challenge 2015.” [Online]. Available: http://aris.me/contents/teaching/data-mining-2015/project/BigDataChallengeData.html

[163] I. F. Akyildiz and W. Wang, “The predictive user mobility profile framework for wireless multimedia networks,” *IEEE/ACM Transactions on Networking (TON)*, vol. 12, no. 6, pp. 1021–1035, 2004.

[164] E. Hossain and V. K. Bhargava, “Link-level traffic scheduling for providing predictive qos in wireless multimedia networks,” *IEEE Transactions on Multimedia*, vol. 6, no. 1, pp. 199–217, 2004.

[165] X. Xing, T. Jing, W. Cheng, Y. Huo, and X. Cheng, “Spectrum prediction in cognitive radio networks,” *IEEE Wireless Communications*, vol. 20, no. 2, pp. 90–96, 2013.

[166] Z. Wei, Q. Zhang, Z. Feng, W. Li, and T. A. Gulliver, “On the construction of radio environment maps for cognitive radio networks,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2013, pp. 4504–4509.

[167] H. B. Yilmaz, T. Tugcu, F. Alagöz, and S. Bayhan, "Radio environment map as enabler for practical cognitive radio networks," *IEEE Communications Magazine*, vol. 51, no. 12, pp. 162–169, 2013.

[168] M. Höyhtyä, A. Mämmelä, M. Eskola, M. Matinmikko, J. Kalliovaara, J. Ojaniemi, J. Suutala, R. Ekman, R. Bacchus, and D. Roberson, "Spectrum occupancy measurements: A survey and use of interference maps," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2386–2414, 2016.

[169] Y. Chen and H.-S. Oh, "A survey of measurement-based spectrum occupancy modeling for cognitive radios," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 848–859, 2016.

[170] K. M. Thilina, K. W. Choi, N. Saquib, and E. Hossain, "Machine learning techniques for cooperative spectrum sensing in cognitive radio networks," *IEEE Journal on selected areas in communications*, vol. 31, no. 11, pp. 2209–2221, 2013.

[171] Z. Khan, J. J. Lehtomäki, L. A. DaSilva, E. Hossain, and M. Latva-Aho, "Opportunistic channel selection by cognitive wireless nodes under imperfect observations and limited memory: a repeated game model," *IEEE Transactions on Mobile Computing*, vol. 15, no. 1, pp. 173–187, 2016.

[172] Y. Saleem and M. H. Rehmani, "Primary radio user activity models for cognitive radio networks: A survey," *Journal of Network and Computer Applications*, vol. 43, pp. 1–16, 2014.

[173] M. Monemi, M. Rasti, and E. Hossain, "Characterizing feasible interference region for underlay cognitive radio networks," in *IEEE International Conference on Communications (ICC)*. IEEE, 2015, pp. 7603–7608.

[174] ——, "On characterization of feasible interference regions in cognitive radio networks," *IEEE Transactions on Communications*, vol. 64, no. 2, pp. 511–524, 2016.

[175] M. Ozger and O. B. Akan, "On the utilization of spectrum opportunity in cognitive radio networks," *IEEE Communications Letters*, vol. 20, no. 1, pp. 157–160, 2016.

[176] F. Akhtar, M. H. Rehmani, and M. Reisslein, "White space: Definitional perspectives and their role in exploiting spectrum opportunities," *Telecommunications Policy*, vol. 40, no. 4, pp. 319–331, 2016.

[177] A. A. Khan, M. H. Rehmani, and M. Reisslein, "Cognitive radio for smart grids: Survey of architectures, spectrum sensing mechanisms, and networking protocols," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 860–898, 2016.

[178] S. H. R. Bukhari, M. H. Rehmani, and S. Siraj, “A survey of channel bonding for wireless networks and guidelines of channel bonding for futuristic cognitive radio sensor networks,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 924–948, 2016.

[179] Z. R. Zaidi and B. L. Mark, “Real-time mobility tracking algorithms for cellular networks based on Kalman filtering,” *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 195–208, 2005.

[180] G. Pappas and M. Zohdy, “Extended Kalman filtering and pathloss modeling for shadow power parameter estimation in mobile wireless communications,” *International Journal on Smart Sensing and Intelligent Systems*, vol. 7, no. 2, pp. 898–924, 2014.

[181] H. Kaaniche and F. Kamoun, “Mobility prediction in wireless ad hoc networks using neural networks,” *Journal of Telecommunications*, vol. 2, no. 1, pp. 95–101, 2010.

[182] NGMN. Next Generation Mobile Networks. [Online]. Available: http://www.ngmn.de/publications/all-downloads/article/ngmn-5g-white-paper.html

[183] I. Malanchini, S. Valentin, and O. Aydin, “Wireless resource sharing for multiple operators: Generalization, fairness, and the value of prediction,” *Elsevier Computer Networks*, vol. 100, pp. 110–123, 2016.

[184] G. P. Fettweis, “The tactile internet: applications and challenges,” *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014.

[185] V. Suryaprakash and I. Malanchini, “Reliability in future radio access networks: from linguistic to quantitative definitions,” in *IEEE/ACM International Symposium on Quality of Service (IWQoS)*, 2016.

[186] N. Singer, “Sharing data, but not happily,” http://www.nytimes.com/2015/06/05/technology/consumers-conflicted-over-data-mining-policies-report-finds.html?_r=0, 2015, the New York Times, [Online; accessed 5-November-2016].

[187] J. Wan, D. Zhang, S. Zhao, L. Yang, and J. Lloret, “Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions,” *IEEE Communications Magazine*, vol. 52, no. 8, pp. 106–113, 2014.

[188] A. Burulitisz, S. Imre, and S. Szabó, “On the accuracy of mobility modelling in wireless networks,” in *Proceedings IEEE ICC*, 2004.

[189] L. Song, D. Kotz, R. Jain, and X. He, “Evaluating location predictors with extensive Wi-Fi mobility data,” in *IEEE INFOCOM*, 2004, pp. 1414–1424.

[190] W. Creixell and K. Sezaki, “Routing protocol for ad hoc mobile networks using mobility prediction,” *International Journal of Ad Hoc and Ubiquitous Computing*, vol. 2, no. 3, pp. 149–156, 2007.

[191] W. Dong, N. Duffield, Z. Ge, S. Lee, and J. Pang, "Modeling cellular user mobility using a leap graph," in *Passive and Active Measurement.* Springer, 2013, pp. 53–62.

[192] Y. Qiao, J. Skicewicz, and P. Dinda, "An empirical study of the multiscale predictability of network traffic," in *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on.* IEEE, 2004, pp. 66–76.

[193] J. Yao, S. S. Kanhere, and M. Hassan, "An empirical study of bandwidth predictability in mobile computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization.* ACM, 2008, pp. 11–18.

[194] P. A. Zandbergen, "Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning," *Transactions in GIS*, vol. 13, no. s1, pp. 5–25, 2009.

[195] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi, "Understanding individual human mobility patterns," *Nature*, vol. 453, no. 7196, pp. 779–782, 2008.

[196] V. A. Siris and D. Kalyvas, "Enhancing mobile data offloading with mobility prediction and prefetching," in *Proceedings ACM MobiArch*, 2012.

[197] O. Osterbo, "Scheduling and capacity estimation in LTE," in *International Teletraffic Congress (ITC)*, 2011.

[198] S. Sesia, I. Toufik, and M. Baker, *LTE: the UMTS long term evolution.* Wiley Online Library, 2009.

[199] T. S. Rappaport *et al.*, *Wireless communications: principles and practice.* Prentice Hall PTR New Jersey, 1996, vol. 2.

[200] S. Makridakis and M. Hibon, "ARMA Models and the Box–Jenkins Methodology," *Journal of Forecasting*, vol. 16, no. 3, pp. 147–163, 1997.

[201] H. W. Lilliefors, "On the kolmogorov-smirnov test for normality with mean and variance unknown," *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[202] M. Ahmed, S. Spagna, F. Huici, and S. Niccolini, "A peek into the future: predicting the evolution of popularity in user generated content," in *ACM WSDM*, Rome, Italy, February 2013, pp. 607–616.

[203] J. D. Hamilton, *Time series analysis.* Princeton university press Princeton, 1994, vol. 2.

[204] R. Pantos and W. May, "HTTP live streaming," *IETF Draft, June*, 2010.

[205] I. Gurobi Optimization, "Gurobi optimizer reference manual," 2016. [Online]. Available: http://www.gurobi.com

[206] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the impact of video quality on user engagement," *ACM SIGCOMM Computer Communication Review*, vol. 41, no. 4, pp. 362–373, 2011.

[207] A. K. Moorthy, L. K. Choi, A. C. Bovik, and G. De Veciana, "Video quality assessment on mobile devices: Subjective, behavioral and objective studies," *IEEE J-STSP*, vol. 6, no. 6, pp. 652–671, 2012.

[208] G. Majid, J. Capka, and R. Boutaba, "Prediction-based admission control for DiffServ wireless internet," in *Proc. IEEE VTC-Fall*, 2003.

[209] P. Koutsakis, M. Vafiadis, and H. Papadakis, "Prediction-based resource allocation for multimedia traffic over high-speed wireless networks," *AEU-International Journal of Electronics and Communications*, 2006.

[210] T. Braun, C. Castelluccia, G. Stattenberger, and I. Aad, "An analysis of the diffserv approach in mobile environments," in *Proc. IQWiM-Workshop*, 1999.

[211] T. Taleb and A. Ksentini, "QoS/QoE predictions-based admission control for femto communications," in *Proc. IEEE ICC*, 2012.

[212] V. Joseph and G. de Veciana, "NOVA: QoE-driven optimization of DASH-based video delivery in networks," in *Proc. IEEE INFOCOM*, 2014.

[213] A. Ashraf, F. Jokhio, T. Deneke, S. Lafond, I. Porres, and J. Lilius, "Stream-based admission control and scheduling for video transcoding in cloud computing," in *Proc. IEEE/ACM CCGrid*, 2013.

[214] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An Open-Source Platform for LTE Evolution and Experimentation," *preprint arXiv:1602.04629*, 2016.

[215] Nuand, "Bladerf software defined radio," http://www.nuand.com, last accessed May 2016.

[216] Ettus Research, "Universal software radio peripheral," http://Ettus.com/, last accessed May 2016.

[217] ETSI, "E-UTRA; Multiplexing and channel coding," *3GPP TS*, vol. 36.212, p. V13, 2016.

[218] S. Kumar, E. Hamed, D. Katabi, and L. Erran Li, "LTE radio analytics made easy and accessible," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, Oct. 2014, pp. 211–222.

[219] X. Xie, X. Zhang, S. Kumar, and L. Erran Li, "piStream: Physical Layer Informed Adaptive Video Streaming Over LTE," in *ACM MobiCom*, Sep. 2015.

[220] ETSI, “E-UTRA; Physical channel and modulation,” *3GPP TS*, vol. 36.211, p. V13, 2016.

[221] R. Falkenberg, C. Ide, and C. Wietfeld, “Client-Based Control Channel Analysis for Connectivity Estimation in LTE Networks,” in *IEEE Vehicular Technology Conference (VTC-Fall)*, Sept 2016, pp. 1–6.

[222] Qualcomm, “Qualcomm eXtensible Diagnostic Monitor,” https://www.qualcomm.com/, last accessed May 2016.

[223] Actix International Limited, “Actix analyzer,” http://www.actix.com, last accessed May 2016.

[224] ascom, “TEMS Investigation,” http://www.ascom.com/, last accessed May 2016.

[225] J. Demel, S. Koslowski, and F. K. Jondral, “A LTE receiver framework using GNU Radio,” *Journal of Signal Processing Systems*, vol. 78, no. 3, pp. 313–320, Jan. 2015.

[226] N. Nikaein, R. Knopp, F. Kaltenberger, L. Gauthier, C. Bonnet, D. Nussbaum, and R. Ghaddab, “OpenAirInterface 4G: an open LTE network in a PC,” in *ACM MobiCom*, 2014.

[227] ETSI, “E-UTRA; Physical layer procedures,” *3GPP TS*, vol. 36.213, p. V13, 2016.

[228] ——, “E-UTRA; Radio Resource Control (RRC); Protocol specification,” *3GPP TS*, vol. 36.331, p. V13, 2016.

[229] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, “A close examination of performance and power characteristics of 4G LTE networks,” in *ACM MobiSys*, Low Wood Bay, Lake District, United Kingdom, June 2012, pp. 225–238.

[230] J. Huang, F. Qian, Y. Guo, Y. Zhou, Q. Xu, Z. M. Mao, S. Sen, and O. Spatscheck, “An in-depth study of LTE: Effect of network protocol and application behavior on performance,” in *ACM SIGCOMM*, Hong Kong, China, August 2013, pp. 363–374.

[231] J. B. Landre, Z. E. Rawas, and R. Visoz, “Lte performance assessment prediction versus field measurements,” in *IEEE PIMRC*, Sep. 2013, pp. 2866–2870.

[232] A. Nikravesh, D. R. Choffnes, E. Katz-Bassett, Z. M. Mao, and M. Welsh, “Mobile network performance from user devices: A longitudinal, multidimensional analysis,” in *Springer PAM*, 2014, pp. 12–22.

[233] A. Nikravesh, H. Yao, S. Xu, D. Choffnes, and Z. M. Mao, “Mobilyzer: An open platform for controllable mobile network measurements,” in *ACM MobiSys*, 2015, pp. 389–404.

[234] Ookla, “Ookla speedtest mobile apps,” http://www.speedtest.net/mobile/, last accessed February 2017.

[235] Y. Xu, Z. Wang, W. K. Leong, and B. Leong, "An end-to-end measurement study of modern cellular data networks," in *Passive and Active Measurement*. Springer, 2014, pp. 34–45.

[236] C. Ide, B. Dusza, and C. Wietfeld, "Performance of channel-aware M2M communications based on LTE network measurements." in *IEEE PIMRC*, 2013, pp. 1614–1618.

[237] Y. Li, C. Peng, Z. Yuan, J. Li, H. Deng, and T. Wang, "Mobileinsight: Extracting and analyzing cellular network information on smartphones," in *ACM Mobicom*, Oct. 2016.

[238] A. Elnashar and M. A. El-Saidny, "Looking at lte in practice: A performance analysis of the lte system based on field test results," *IEEE Vehicular Technology Magazine*, vol. 8, no. 3, pp. 81–92, Sep. 2013.

[239] N. Becker, A. Rizk, and M. Fidler, "A measurement study on the application-level performance of lte," in *IFIP Networking Conference*, June 2014, pp. 1–9.

[240] M. Jovanovic, M. K. Karray, and B. Blaszczyszyn, "QoS and network performance estimation in heterogeneous cellular networks validated by real-field measurements," in *ACM PE-WASUN*, 2014, pp. 25–32.

[241] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, "Streaming over 3g and lte: How to save smartphone energy in radio access network-friendly way," in *ACM MoVid*, May 2013, pp. 13–18.

[242] M. Palola, M. Matinmikko, J. Prokkola, M. Mustonen, M. Heikkilä, T. Kippola, S. Yrjölä, V. Hartikainen, L. Tudose, A. Kivinen, J. Paavola, and K. Heiska, "Live field trial of licensed shared access (LSA) concept using LTE network in 2.3 GHz band," in *IEEE DYSPAN*, Apr. 2014, pp. 38–47.

[243] W. Li, R. K. P. Mok, D. Wu, and R. K. C. Chang, "On the accuracy of smartphone-based mobile network measurement," in *IEEE INFOCOM*, Apr. 2015, pp. 370–378.

[244] Motorola, "MotoG LTE," http://www.devicespecifications.com/en/model/10da2ca3, last accessed November 2016.

[245] Huawei, "P8 Lite," http://www.devicespecifications.com/en/model/b95e33c3, last accessed November 2016.

[246] ZTE, "Blade A452," http://www.devicespecifications.com/en/model/a44f38ba, last accessed November 2016.

[247] MartinGarcia, Luis, "TCPdump," http://www.tcpdump.org/, last accessed November 2016.

[248] Wireless Open Access Research Platform, "802.11 reference design for WARP v3," http://warpproject.org/trac/wiki/802.11, last accessed November 2016.

[249] ETSI, “E-UTRA; Medium Access Control (MAC) protocol specification,” *3GPP TS*, vol. 36.321, p. V13, 2016.

[250] A. Furno, M. Fiore, R. Stanica, C. Ziemlicki, and Z. Smoreda, “A tale of ten cities: Characterizing signatures of mobile traffic in urban areas,” *IEEE Transactions on Mobile Computing*, 2016.

[251] A. Furno, M. Fiore, and R. Stanica, “Joint spatial and temporal classification of mobile traffic demands,” in *IEEE INFOCOM*, April.

[252] H. Wang, F. Xu, Y. Li, P. Zhang, and D. Jin, “Understanding mobile traffic patterns of large scale cellular towers in urban environment,” in *Proceedings of the 2015 ACM Conference on Internet Measurement Conference*. ACM, 2015, pp. 225–238.

[253] J. Ding, X. Liu, Y. Li, D. Wu, D. Jin, and S. Chen, “Measurement-driven capability modeling for mobile network in large-scale urban environment,” in *Mobile Ad Hoc and Sensor Systems (MASS), 2016 IEEE 13th International Conference on*. IEEE, 2016, pp. 92–100.

[254] M. Z. Shafiq, L. Ji, A. X. Liu, J. Pang, S. Venkataraman, and J. Wang, “A first look at cellular network performance during crowded events,” in *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1. ACM, 2013, pp. 17–28.

[255] R. Keralapura, A. Nucci, Z.-L. Zhang, and L. Gao, “Profiling users in a 3G network using hourglass co-clustering,” in *ACM MobiCOM*, 2010, pp. 341–352.

[256] N. Bui, F. Michelinakis, and J. Widmer, “Fine-grained LTE Radio Link Estimation for Mobile Phones,” in *submitted to IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2017.

[257] A. Gerber, J. Pang, O. Spatscheck, and S. Venkataraman, “Speed testing without speed tests: estimating achievable download speed from passive measurements,” in *ACM IMC*, Melbourne, Australia, November 2010, pp. 424–430.

[258] F. Ricciato, F. Vacirca, and M. Karner, “Bottleneck Detection in UMTS via TCP Passive Monitoring: A Real Case,” in *ACM CoNEXT*, Toulouse, France, October 2005, pp. 211–219.

[259] P. Svoboda and F. Ricciato, “Analysis and detection of bottlenecks via TCP footprints in live 3G networks,” in *IFIP WiOPT*, Hammammet, Tunisia, April 2008, pp. 37–42.

[260] K. Lai and M. Baker, “Measuring link bandwidths using a deterministic model of packet delay,” in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM ’00. New York, NY, USA: ACM, 2000, pp. 283–294. [Online]. Available: http://doi.acm.org/10.1145/347059.347557

[261] C. Dovrolis, P. Ramanathan, and D. Moore, “Packet-dispersion techniques and a capacity-estimation methodology,” *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963–977, December 2004.

[262] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Sanadidi, “CapProbe: a simple and accurate capacity estimation technique,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 67–78, October 2004.

[263] “The network simulator - ns-3,” http://www.nsnam.org/, last accessed September 2015.

[264] “LENA - ns-3 LTE module,” http://lena.cttc.es/manual/, last accessed September 2015.