UNIVERSITÀ DI PISA

Scuola di Dottorato in Ingegneria “Leonardo da Vinci”

Corso di Dottorato di Ricerca in Ingegneria dell’informazione

Tesi di Dottorato di Ricerca

Resource Allocation in LTE Advanced for QoS and Energy Efficiency

Autore:
Daniele Migliorini [illegible]

Relatori:

Prof. Luciano Lenzini

Ing. Giovanni Stea

Anno 2012

A Veronica

Ai miei genitori

Abstract:

Long Term Evolution (LTE) and LTE-Advanced (LTE-A) are establishing themselves as the new standard of 4G cellular networks in Europe and in several other parts of the world. Their success will largely depend on their ability to support Quality of Service for different types of users, at reasonable costs. The quality of service will depend on how effectively the cell bandwidth is shared among the users. The cost will depend – among many other factors – on how effectively we exploit the cell capacity. Being able to exploit bandwidth efficiently postpones the time when network upgrades are required. On the other hand, operation costs also depend on the energy efficiency of the cellular network, which should avoid wasting power when few users are connected.

As for bandwidth efficiency, the recent LTE/LTE-A standards introduced MIMO (Multiple Input Multiple Output) transmission modes, which allow both reliability and efficiency to be increased. MIMO can increase the throughput significantly. In a MIMO system, the selection of the MIMO transmission modes (whether Transmission Diversity, Spatial Multiplexing, or Multi-User MIMO) plays a key feature in determining the achievable rate and the error probability experienced by the users. MIMO-unaware scheduling policies, which neglect the transmission mode selection problem, do not perform well under MIMO. In the current literature, few MIMO-aware LTE-A scheduling policies have been designed. However, despite being proposed for LTE-A, these solutions do not take into account some constraints inherent to LTE-A, hence leading to unfeasible allocations. In this work, we propose a new framework for Transmission Mode Selection and Frequency-Domain Packet Scheduling, which is compliant with the constraints of the LTE-A standard. The resource allocation framework accommodates real-time requirements and fairness on demand, while the bulk of the resources are allocated in an opportunistic fashion, i.e. so as to maximize the cell throughput. Our results show that our proposal provides real-time connections with the desired level of QoS, without utterly sacrificing the cell throughput.

As far as energy efficiency is concerned, we studied the problem of minimizing the RF power used by the eNodeB, while maintaining the same level of service for the users. We devised a provisioning framework that exploits the Multicast/Broadcast over a Single Frequency Network (MBSFN) mechanism to deactivate the eNodeB on some Transmission Time Intervals (TTI), and computes the minimum-power activation required for guaranteeing a given level of service. Our results show that the provisioning framework is stable, and that it allows significant savings with respect to an always-on policy, with marginal impact on the latency experienced by the users.

Sommario:

Long Term Evolution (LTE) e LTE-Advanced (LTE-A) si stanno affermando come il nuovo standard delle reti cellulari di quarta generazione. Il successo di queste tecnologie dipenderà, in larga parte dalla loro capacità di offrire qualità del servizio a differenti tipi di utenti a costi ragionevoli. La qualità del servizio dipende da come effettivamente viene suddivisa la banda tra gli utenti. Il costo dipende –tra molti altri fattori- da come effettivamente si sfrutta la capacità di cella. Riuscire a sfruttare la banda in maniera efficiente rimanda il momento in cui è necessario un upgrade dell'infrastruttura di rete. D'altra parte, i costi dipendono anche dall'efficienza energetica della rete cellulare che dovrebbe evitare sprechi di energia quando ci sono pochi utenti connessi.

Riguardo all'efficienza di banda, le recenti versioni degli standard LTE/LTE-A introducono il concetto di modo trasmissivo ad antenne multiple (MIMO) che permette di incrementare sia l'affidabilità della trasmissione sia l'efficienza. Queste tecniche incrementano il throughput in maniera significativa. In un sistema MIMO la selezione del modo trasmissivo (tra i cui Transmission Diversity, Spatial Multiplexing, o Multi-User MIMO) svolge una funzione chiave nel determinare il throughput e la probabilità di errore sperimentate dall'utente finale. Gli scheduler che non tengono in considerazione le tecniche MIMO (MIMO-unware) e il problema della selezione del corretto modo trasmissivo non offrono buone prestazioni perché non sfruttano tale diversità. In letteratura sono presenti pochi lavori dove si propongono algoritmi di scheduling MIMO-aware. Tuttavia nonostante questi lavori siano proposti per LTE-A non tengono però in considerazione alcuni importanti vincoli presenti nello standard che portano ad allocazioni di risorse non applicabili ad LTE-A. In questa tesi si propone una nuova framework per la selezione della modalità trasmissiva MIMO e un algoritmo di scheduling nel dominio della frequenza compatibile con i vincoli dello standard LTE-A. Questa framework gestisce traffici real-time e fairness su domanda mentre il resto delle risorse sono allocate in maniera prettamente opportunistica massimizzando così il throughput di cella. I risultati mostrano che la nostra proposta offre, ai traffici real-time, il livello desiderato di qualità del servizio senza sacrificare troppo il throughput di cella.

Per quanto riguarda l'efficienza energetica, abbiamo affrontato il problema di minimizzare la potenza RF usata dalla cella mantenendo lo stesso livello di servizio agli utenti. Abbiamo messo a punto una framework di provisioning che sfrutta il meccanismo Multicast/Broadcast over a Single Frequency Network (MBSFN) per disattivare la cella in alcuni Transmission Time Intervals (TTI) e calcola le attivazioni a potenza minima richieste per garantire un certo livello di servizio. I risultati mostrano che la framework di provisioning è stabile e offre un significativo risparmio con un impatto marginale sulla latenza sperimentata dagli utenti.

CONTENTS

Contents 9

LIST OF FIGUREs 11

LIST OF TABLEs 13

1 INTRODUCTION 15

2 LTE Technology 18

2.1 3gpp Releases 18

2.1.1 Improvements 19

2.1.2 System architecture 20

2.1.3 Protocols 22

2.1.4 LTE Advanced 23

2.1.5 Physical Layer Details 24

2.1.5.1 Access schemes 24

2.1.5.2 MIMO modes in LTE 26

3 Proposed Scheduler 29

3.1 Introduction 29

3.2 LTE-A Resource allocation and constraints 29

3.3 System Model and Throughput-Optimal Scheduling Problem 31

3.4 Polynomial-time Opportunistic Scheduling 36

3.4.1 TMS algorithms 37

3.4.2 Allocation function 39

3.5 QoS and Fairness Constraints 42

3.5.1 QoS Step 43

3.5.2 Fairness Step 45

3.5.3 Efficiency Step 46

3.6 Performance evaluation 47

3.6.1 Simulator 47

3.6.1.1 OMNeT++ 47

3.6.1.2 LTE OMNeT++ Embodiment 47

3.6.2 Schedulers 48

3.6.3 Traffic 49

3.6.4 Metrics 50

3.6.5 System Model 51

3.6.6 Results 52

3.6.6.1 QoSMuMIMO Tuning 52

3.6.6.2 Comparison with other scheduler 57

3.7 Related Work 62

4 Energy Aware provisioner 64
4.1 Introduction 64
4.2 MBSFN patterns in LTE-Advanced 64
4.3 System model and problem formulation 65
4.4 Approximating the optimal solution for known traffic demands 67
4.4.1 Throughput-oriented heuristic 69
4.4.2 Predicting the traffic demand 73
4.4.3 Distributing RBs among active frames 74
4.5 Making decisions on a per-TTI basis 75
4.6 Results 77
4.6.1 Simulated scenarios 77
5 Conclusions 82
6 REFERENCES 83

LIST OF FIGURES

Figure 2-1 3GPP Groups 18
Figure 2-2 3GPP Long Term Evolution overview 19
Figure 2-3: LTE Evolved Packet System 21
Figure 2-4: LTE Protocol Stack 22
Figure 2-5: OFDMA 25
Figure 2-6: SC-FDMA 25
Figure 2-7: LTE frame structure 26
Figure 3-1: the Rate Matrix 32
Figure 3-2: Mathematical model for the MTAP 35
Figure 3-3: The resource allocation framework 36
Figure 3-4: K-max algorithm pseudo-code 38
Figure 3-5: Pseudo-code of MaxGain algorithm 41
Figure 3-6: The enhanced allocation function 42
Figure 3-7: Tagger function pseudo code 43
Figure 3-8: Selector function pseudo code 44
Figure 3-9: Fairness step pseudo code 46
Figure 3-10: Average MOS of UE with different fairness settings 54
Figure 3-11: Average delay of gaming traffic with different fairness settings 54
Figure 3-12: Average delay of Video on Demand traffic with different fairness settings 55
Figure 3-13: Average Throughput of full buffer UE with different fairness settings 56
Figure 3-14: Transmission mode usage 57
Figure 3-15: Cell throughput with different scheduler 57
Figure 3-16: Average Full buffer traffic throughput with different scheduler 58
Figure 3-17: Average MOS of Voice over IP traffic with different scheduler 59
Figure 3-18: Average delay of gaming traffic with different scheduler 60
Figure 3-19: Average delay of Video on Demand traffic with different scheduler... 60
Figure 3-20: Average delay of Video on Demand traffic with different scheduler (rescaled) 61
Figure 4-1: LTE-A – Allowed MBSFN subframes 65
Figure 4-2: Power model 66
Figure 4-3: Pseudo-code for the throughput-oriented heuristic 72
Figure 4-4. Pseudo-code for the dynamic algorithm 76
Figure 4-5 Depleted power with increasing network load 78
Figure 4-6 Average delay with increasing network load 79
Figure 4-7 Cell Throughput with 2 MB/s of network load 80
Figure 4-8 Depleted power with increasing activation time of the provisioner........ 80
Figure 4-9 Average delay with increasing activation time of the provisioner 81

LIST OF TABLES

Table 1-1: LTE versus LTE Advanced 23
Table 2-1: Number of bits per CQI 33
Table 2-2: Configuration of traffic models 50
Table 2-3: System model parameters 51
Table 2-4: Simulated scenario 52
Table 2-5: Urgency settings for each traffic class 53
Table 2-6: Fairness step settings 53
Table 3-1: Number of UE for each offered load 78

1 INTRODUCTION

The Long Term Evolution (LTE) of the Universal Mobile Telecommunication System (UMTS) [14]is gaining progressive hold as an access network for Internet services, thanks for its foreseen near-ubiquitous coverage and high bandwidth. At the moment of writing, the first experimental setups of LTE networks are being deployed all over Europe, including Italy. LTE promises higher bandwidth (up to 300 Mbps in the downlink and 75 in the uplink), which makes this technology suitable for ubiquitous Internet access (besides for supporting voice). Due to its coverage, regulated access and security, LTE is also a serious candidate for serving as an infrastructure for *machine to machine* (M2M) communications. In the near future, in fact, intelligent devices will be endowed with communication capabilities, to improve the quality of living in everyday life. Examples of such communications are those arising from *smart grids*, i.e. electrical power grids provided with the intelligence to coordinate a distributed power supply, serving appliances which can schedule their jobs based on price and power availability. All this intelligence requires diffused, wide-scale communication capabilities. Another example is vehicular communications, where single vehicles communicate with a central infrastructure to reduce road dangers, accidents and congestion.
Traffics generated by all the above applications need to coexist on a single infrastructure. Therefore, LTE networks need to be provided with *quality of service (QoS),* i.e., the ability to give different treatment to traffic of different applications. LTE has native QoS differentiation support. Traffic is classified into four *traffic classes,* namely *Conversational*, *Interactive*, *Streaming*, *Background*, whose packet can be treated differently. Although the LTE network is composed by many blocks (i.e., a wired part and a wireless interface), the part where algorithms for QoS differentiation are mostly required is the wireless one, i.e. the e-UTRAN interface in a network cell. In the latter, a central base station or eNodeB shares radio resources among a number of User Equipments (UEs), i.e. handheld devices, laptops or home gateways, using Orthogonal Frequency Division Multiplexing Access (OFDMA) in the downlink. From a logical standpoint, on each Transmission Time Interval (TTI, 1ms), the eNodeB allocates a time/frequency frame of resource blocks (RBs) to the UEs. RBs carry a variable number of bits to an UE, depending on the *transport block size* (TBS) selected by the eNodeB. The latter is chosen based on the Channel Quality Indicator (CQI) advertised by the UE. The scheduling algorithm at the eNode-B, therefore, plays a crucial role into QoS differentiation. The latter should be able to pick those UEs whose channel quality is the highest, so as to exploit the cell bandwidth efficiently, something which is called *opportunism*. On the other hand, since the channel quality may depend on physical details of the receiver and, mostly, on the distance between the eNodeB and the UE, and on the interference generated by nearby cells (which is also location-dependent), only pursuing opportunism as a resource allocation criterion may lead to gross unfairness, where some UEs are starved whereas others hog the whole channel bandwidth. Therefore, opportunism has to be balanced with fairness. Moreover, packets with deadline constraints (e.g., voice, gaming, video) should be transmitted by their deadline, with a comparatively minor attention to opportunism. Striking a good balance between deadlines, fairness and opportunism is already tough enough per se. A last comment, though by no means the least concern, is

related to complexity. Schedules must be built in a TTI time, and may include hundreds of UEs. Therefore, non-polynomial algorithms are out of the picture, and relatively simple polynomial ones should be preferred.
In the endless race for capacity of cellular systems, *Multiple-Input Multiple-Output* (*MIMO*) transmissions have been introduced. LTE (Release 8) allows MIMO in the downlink direction, whereas the more recent Release 10, which goes by the commercial name of *LTE-Advanced (LTE-A)*, allows it for both the downlink and the uplink. MIMO transmissions exploit spatial diversity to send more than one stream to a user, or a pair thereof, on the same RB. Three MIMO transmission modes are possible: *transmit diversity* (*TD*), where (with reference to the downlink direction) the same codeword is sent *twice* to the UE, thus increasing the likelihood of correct decoding and, indirectly, allowing higher-order modulations (e.g. 64QAM) to be exploited; *spatial multiplexing* (S-MUX), where two *different* codewords are sent to the same UE, thus doubling its rate; *multi-user MIMO* (MU-MIMO), where two spatially separated UEs are coupled on the same RB. TD and S-MUX are called *single-user* MIMO (SU-MIMO) modes. LTE-A allows an eNodeB to decide which MIMO mode to use for a given UE (or pair thereof) dynamically, on a per-TTI basis.
It is a fact that MIMO transmissions increase the cell capacity. However, resource scheduling under MIMO is considerably more involved. In fact, besides deciding which RB goes to whom, a *transmission mode* has to be selected for each UE (or pair thereof). Transmission mode selection plays a role on how many bits you can fit into an RB, hence *transmission mode selection* and *frequency-domain packet scheduling* (i.e., RB allocation) are not independent.
As a first contribution of this thesis, we investigate resource allocation in the downlink of a MIMO-capable LTE-A system. Our purpose is to provide a scheduling framework which accommodates different types of traffic, *and* exploits radio resources efficiently. We first show that computing the *maximum throughput allocation* in a MIMO-enabled LTE cell is practically unfeasible given the number of UEs involved and the scheduling timescale. Hence, we propose to separate the resource allocation problem into two sub-problems, namely *transmission mode selection (TMS)* and *frequency-domain packet scheduling* (*FDPS*). For the TMS problem, we propose a tunable polynomial-time algorithm. For the FPDS sub-problem, we propose a modular framework that allows different types of services to be accommodated *on demand*. The basic block is an opportunistic allocation, which maximizes the throughput. Other blocks can be added if traffics with fairness (i.e., minimum throughput) or QoS (i.e., deadline) constraints are to be considered. We show via simulation that the opportunistic version of the FPDS achieves near-optimal cell throughput, and that adding QoS and fairness only entails a modest throughput loss. To the best of our knowledge, no previous work covers multi-service scheduling under MIMO constraints. Those which consider single aspects (e.g., only throughput maximization, or fairness, under MU-MIMO) often rely on simplifications and idealizations that make them unsuitable for the LTE-A environment. Some either ignore MU-MIMO (e.g. 16), or assume arbitrary, out-of-standard MU-MIMO UE pairings in the same TTI (e.g. 17), others neglect the TMS problem (which greatly influences the cell throughput) by forcing *a priori* a single transmission mode (e.g. [40][41]).
As a second contribution of this thesis, we study the *energy efficiency* of resource allocation in the above settings. There has been a growing interest in the energy

consumption aspects of computer networks in the last years. In the cellular network environment, this has been largely motivated by the operators' need to cut the energy bill in order to save on the operational costs. It is a fact that cellular networks (and others beside it) are dimensioned for peak demand, and are underutilized in off-peak hours (e.g., at night). LTE's flexible, millisecond-based frame structure makes it possible to selectively deactivate transmissions on some TTIs at times of low load without utterly sacrificing responsiveness. The *discontinuous transmission (DTX)* mechanism allows the eNodeB to communicate to its associated UEs a *pattern* of future activations (e.g., a 40ms superframe), marking some frames as *inactive*. During these, the eNodeB will be switched off and no communication will take place. To complicate the scenario, there are frames where UEs are supposed to receive pilot signals (e.g., synchronization, paging, etc.), hence not all the frames can be switched off. Furthermore, due to the fact that OFDMA symbols are reserved for pilot signals, the number of OFDMA symbols which are available for UE transmission, i.e. the *frame capacity*, varies among frames.
The second contribution of this thesis is a *provisioning algorithm* that computes a frame activation pattern in a superframe, so as to carry an expected load at the minimum power. The provisioning algorithm relates to the scheduler, which is activated only during *on* frames. Depending on the bandwidth efficiency of the scheduler, the energy efficiency will vary as well. We present the minimum-power allocation as an optimization problem, and we propose an algorithm that finds a good suboptimal solution given the load. The provisioning algorithm is tested in conjunction related to the MIMO scheduler, and proved to be stable. Furthermore, we show that the energy saving is indeed significant.
There has been scant little work on DTX so far. The EARTH project of the EU 6th Framework Programme has dealt with the issue, but its publications are appearing as we write down the thesis, and only preliminary studies are available [41], for instance, shows the saving potentials for metropolitan areas, but does not present any algorithm.
This thesis is organized as follows: In chapter 2 we describe the LTE technology, the proposed MIMO-aware scheduler is presented in chapter 3 and evaluated in section 3.6. In chapter 4 we presents the energy aware provisioner while in section 4.6 we show the results of this algorithm. Conclusion are drawn in chapter 5.

2 LTE TECHNOLOGY

In this chapter we briefly introduce the Long Term Evolution (LTE) technology, pointing out the motivations that lead to its development, its base characteristics and the future evolutions. Then we provide some details on the physical layer, considering in particular the Multiple Input Multiple Output (MIMO) transmission modes used within LTE, which are one of the key topics of the present work.

2.1 3gpp Releases

In recent years the number of mobile subscribers has increased tremendously and voice communication has become mobile in a massive way. At the same time data usage has grown fast and mobile devices are now used for a wide range of other applications like web browsing, video streaming and online gaming. Beside cell phones, also notebooks, personal digital assistants and other hand-held devices are now part of the mobile environment, leading to a heterogeneous population of User Equipment (UEs) with different needs.
Within such a scenario, the demand for an ubiquitous and broadband Internet access is constantly increasing, and mobile communication systems have to provide evolved technologies to support both voice and data traffic.

The Third Generation Partnership Project (3GPP) [1] is a collaboration among international groups of telecommunication associations, founded to define globally applicable standards for next-generation mobile networks (
Figure 2-1).
Established in 1998, the 3GPP has developed several standards, named *releases,* to support the increasing needs of mobile communications (Figure 2-2).

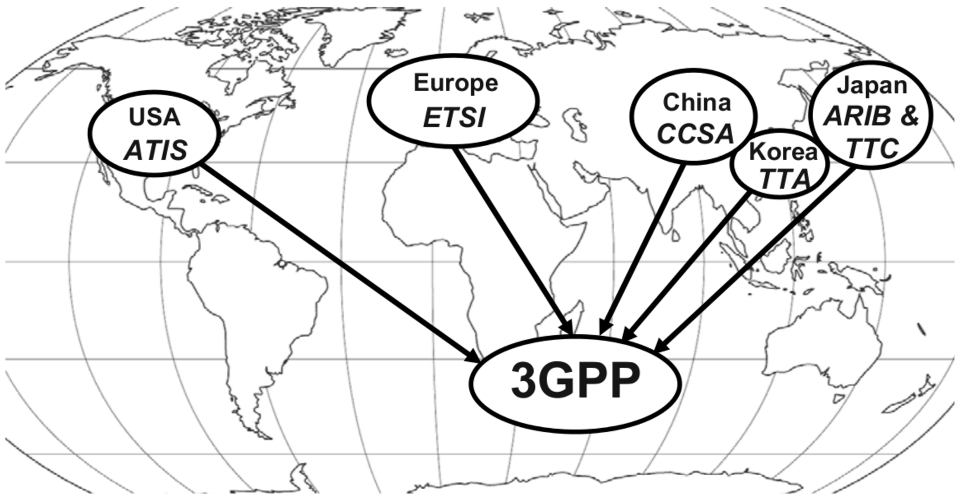


Figure 2-1 3GPP Groups

Among these releases, LTE [2] represents a major step towards the support of heterogeneous traffics and different kind of UEs. Based on LTE, LTE Advanced [3]

is the further step of the 3GPP Release standardization process to meet the requirements imposed by the International Telecommunication Union (ITU) [4] for fourth-generation (4G) networks.
Here both LTE and LTE Advanced are briefly described. First we consider the improvements introduced by LTE with respect to legacy 3GPP releases. Then we present the system architecture and the protocol stack. Finally we underline the enhancements introduced by LTE Advanced.

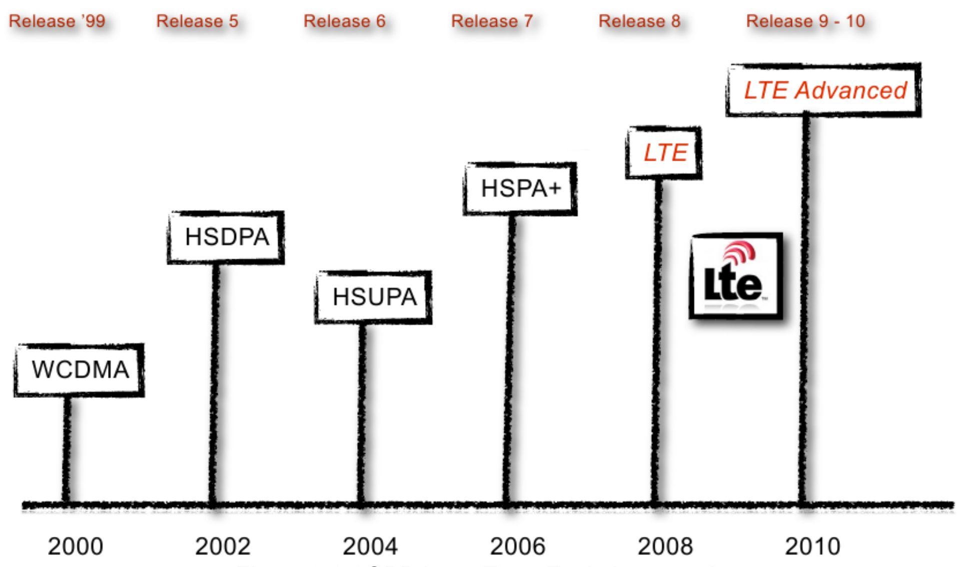


Figure 2-2 3GPP Long Term Evolution overview

2.1.1 Improvements

The first LTE release has been developed as the natural evolution of High Speed Packet Access (HSPA) in order to ensure the competitive- ness of third-generation (3G) technologies for the next years. To reach this objective, LTE introduces several improvements with respect to the previous releases. Such improvements are summarized below.

- Data rate: it is possible to reach a peak data rate of 300 Mbps in downlink and 75 Mbps in uplink, using a 20 MHz bandwidth.
- Spectrum flexibility: scalable bandwidths from 1.25 to 20 MHz are supported.
- Architecture simplification: the elements within the UMTS Terrestrial Radio Access Network (UTRAN) have been reduced, thus leading to a flat architecture.
- Enhanced support for mobility:
  - optimal support for slow-moving users (0 - 15 km/h)
  - high performance for speeds between 15 - 150 km/h
  - high speeds (up to 350 km/h) are also supported
- Reduced latency: the one-way transit time between a packet being available at the IP layer in either the UE or radio access network and the

availability of this packet on the counterpart is less than 5 ms. Also Control Plane latency is reduced, being less than 100 ms.

- Enhanced support for end-to-end Quality of Service (QoS): an all-IP paradigm is used to deal with different traffic flows. Voice traffic is served as VoIP (Voice over IP) and the call quality should be at least the same as in UMTS circuit switched networks.
- Inter-working: inter-working with existing legacy 3GPP systems and non-3GPP systems is ensured, with little handover interruption time (between 300 and 500 ms).

These improvements rely on some enabling technologies which were not considered in the previous 3GPP releases. Among others, it is worth mentioning the following two aspects:

- the adoption of new medium access schemes:
  - Orthogonal Frequency Division Multiple Access (OFDMA) for downlink.
  - Single-Carrier Frequency Division Multiple Access (SC-FDMA) for uplink.
- the inclusion of some advanced MIMO techniques:
  - Spatial Multiplexing
  - Transmit Diversity
  - Multi-User MIMO

Both these aspects are described in 2.1.5

2.1.2 System architecture

Figure 1.3 shows a simplified scheme of the LTE network architecture [8] , referred as Evolved Packet System (EPS).

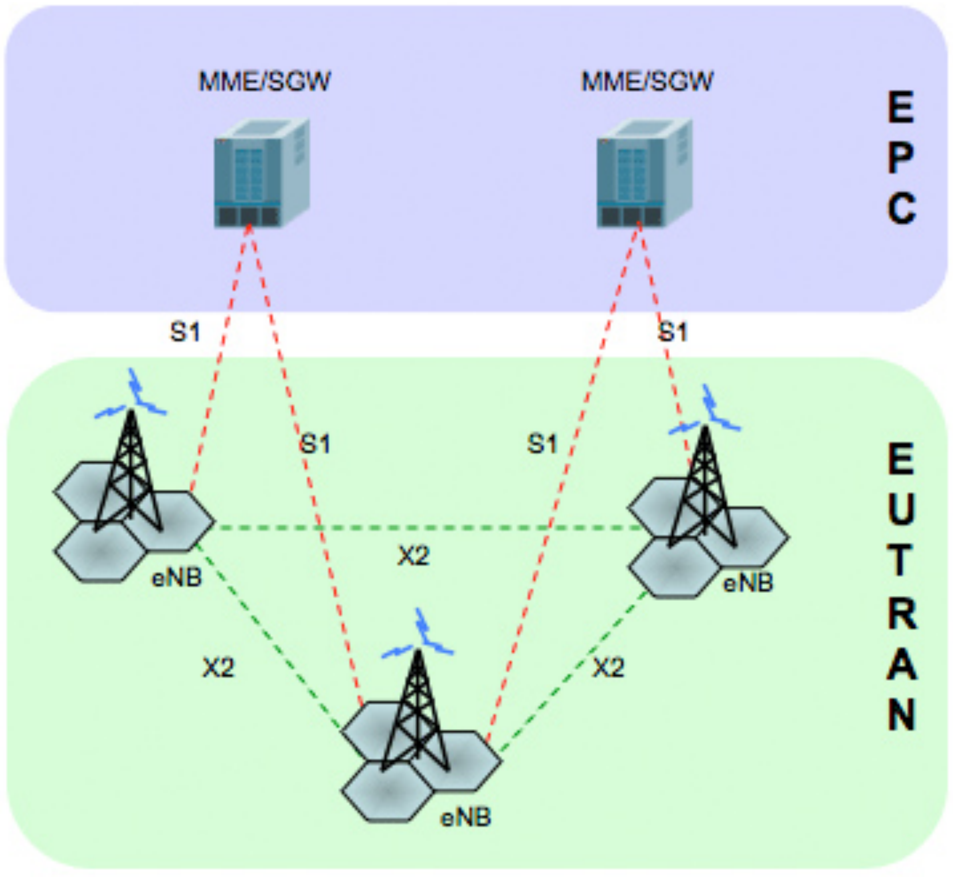


Figure 2-3: LTE Evolved Packet System

The EPS is composed by two parts: access network and core network.

- The *Evolved UMTS Terrestrial Radio Access Network* (E-UTRAN) is the access network and it is composed by two different types of node:
  - User Equipment, that is the devices used by mobile users for communication.
  - Enhanced Node B (eNodeB), that is the cell base station, which is in control of all radio related functions in the fixed part of the system.
- The *Evolved Packet Core* (EPC) is the core network and it is mainly composed by two entities:
  - *Mobility Management Entity* (MME), which takes care of the issues related to user identification, security and mobility management.
  - *System Architecture Evolution Gateway* (SAE-GW), which is the data interface between the access network and external networks.

EPS nodes are connected among them through optical fiber links, in order to exchange both data and control information. The following interface are defined:

- X2 interface - between different eNodeBs.

- S1 interface - between the eNodeBs and the EPC.

2.1.3 Protocols

LTE is composed by several protocols, each performing specific functions. The LTE protocol stack [8] is shown in Figure 2-4.

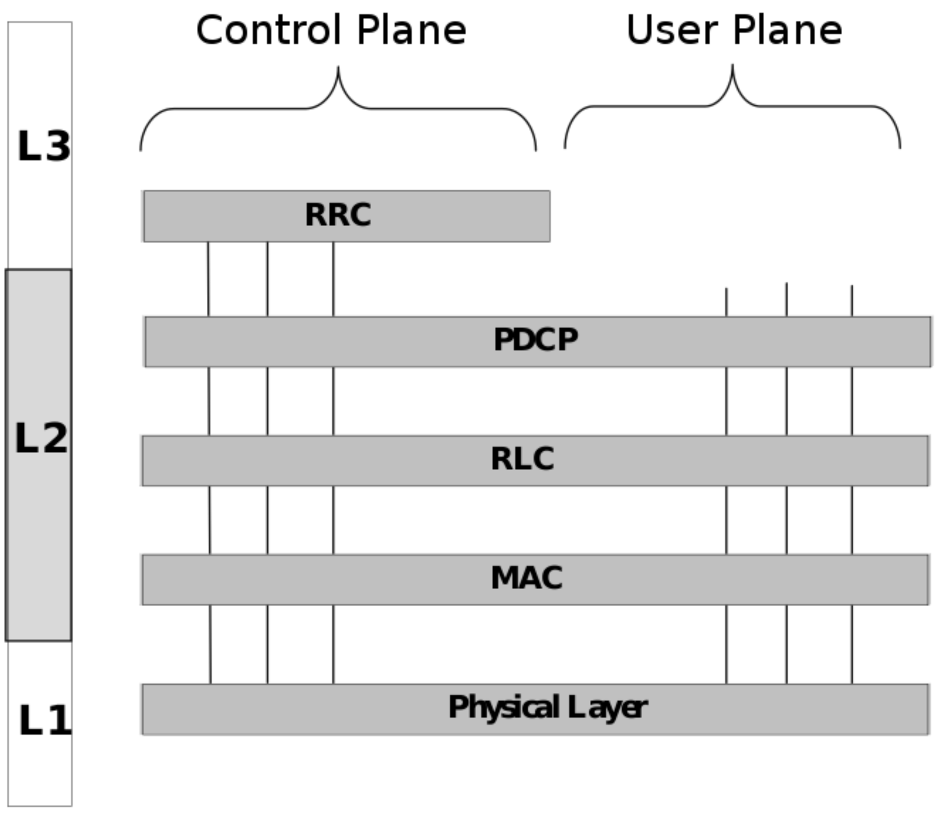


Figure 2-4: LTE Protocol Stack

The stack distinctly manages information belonging to two different planes: the Control Plane and the User Plane. The Control Plane is composed by all control messages. The main control layer is RRC (Radio Resource Control), which manages the major part of control information exchanged between UE and eNodeB. The User Plane, instead, is composed by user data packets coming from and going to the IP network layer.
Data traffic is managed within the LTE stack by the layers listed below.

- Packet Data Convergence Protocol (PDCP) - This layer performs header compression/decompression on IP packets and ciphering/deciphering operations.
- Radio Link Control (RLC) - Depending on the RLC mode used1, this layer may perform error detection and correction with Automatic Repeat reQuest

(ARQ) procedures, concatenation and segmentation, in-sequence delivery and duplicate detection. It is also possible to have a timer-based mechanism to discard packets.

- Medium Access Control (MAC) - This layer performs multiplexing/demultiplexing of RLC Protocol Data Units (PDUs) in MAC PDUs and it is responsible for dynamic scheduling of users, both in downlink and in uplink. Within scheduling operations there is also the transport format selection, including transmission mode selection and link adaptation2. The interface between MAC and the physical layers is composed by Hybrid-ARQ (H-ARQ), which provides a mechanism for error correction by combining different retransmissions.
- Physical layer (PHY) - This layer provides the actual air interface to the radio channel. It is responsible for modulation and coding and performs antenna allocations according to the scheduling decisions. This layer encodes the data coming from higher layers into *codewords* for the transmission over the wireless channel. More details on the physical layer are given in 2.1.5.

A complete description of LTE protocols would require considerably more space and falls outside the scope of the present work. Further information about the LTE stack functionalities can be found in [8] or referring to the standard [83].

2.1.4 LTE Advanced

Release 10, called LTE Advanced, has been developed by 3GPP in order to fulfill the International Mobile Telecommunications Advanced (IMT-Advanced) requirements defined by ITU for 4G networks.
LTE Advanced is an evolution of Release 8 more than a new ver- sion3. It is backward compatible with the LTE technology and LTE Advanced networks are supposed to appear to legacy LTE terminals as standard LTE networks. The main differences between the two releases are summarized in Table 2-1.

	LTE (Rel. 8)	LTE Advanced (Rel. 10)
Peak Data Rate DL	300 Mbps	1 Gbps
Peak Data Rate UL	75 Mbps	500 Mbps
Maximum Bandwidth	20 MHz	100 MHz
DL MIMO Support	up to 4 layers	up to 8 layers

Table 2-1: LTE versus LTE Advanced

In addition to the above differences, LTE Advanced introduces also
the following features:

- The use of Relay nodes to extend cell coverage or use high MCSs also for border cell users. Relay nodes exchange information through the wireless channel with both the Donor eNodeB (DeNB) and UEs.

- The introduction of enhanced multi-antenna techniques, with Coordinated Multi-Point (CoMP) transmission and reception, and enhancements in Multi-User MIMO.
- The use of Distributed Antenna Systems (DAS), which form a network of spatially separated antenna nodes within the cell area, connected to the eNodeB by optical fiber links.

2.1.5 Physical Layer Details

To ease the comprehension of the work proposed in the following chapters, here we provide some details on LTE physical layer, mainly focusing our attention on the access schemes and the MIMO techniques used.

2.1.5.1 Access schemes

As we briefly noticed in 2.1.1, LTE uses OFDMA in downlink and SC-FDMA in uplink. Both access schemes can be viewed as multi-user versions of the Orthogonal Frequency Division Multiplexing (OFDM) digital modulation scheme.

3 Release 9 has some minor innovations with respect to the previous release. 4 Within this table, *DL* stands for downlink, while *UL* stands for uplink. By *MIMO layer* we mean a spatial channel In OFDM a large number of closely-spaced orthogonal subcarriers are used to carry data. Data are divided into several parallel streams sent over the various subcarriers and each subcarrier is modulated independently.

In OFDMA and SC-FDMA multiple access is achieved by assigning subsets of subcarriers to individual UEs. The distinguishing feature of SC-FDMA is that it leads to a single-carrier transmit signal, in contrast to OFDMA which is a multi-carrier transmission scheme (Figure 2-5 and Figure 2-6).

Thanks to the single-carrier structure, a prominent advantage of SC-FDMA over OFDMA is that its transmit signal has a lower Peak-to- Average Power Ratio (PAPR), thus leading to a lower power consumption: for this reason this access scheme has been adopted in uplink, where UE’s battery duration is crucial. On the other hand, OFDMA is used in the downlink direction to minimize receiver complexity and to enable frequency domain scheduling with flexibility in resource allocation. In fact the the multi-carrier nature of OFDMA allows the scheduler to allocate non contiguous frequency chunks to a given UE, while in SC-FDMA the allocation is continuous as only one symbol is transmitted at a time.

The modulation schemes available in LTE are QPSK (Quadrature Phase Shift Keying), 16QAM (Quadrature Amplitude Modulation) and 64QAM. For each scheme several coding rate are available.

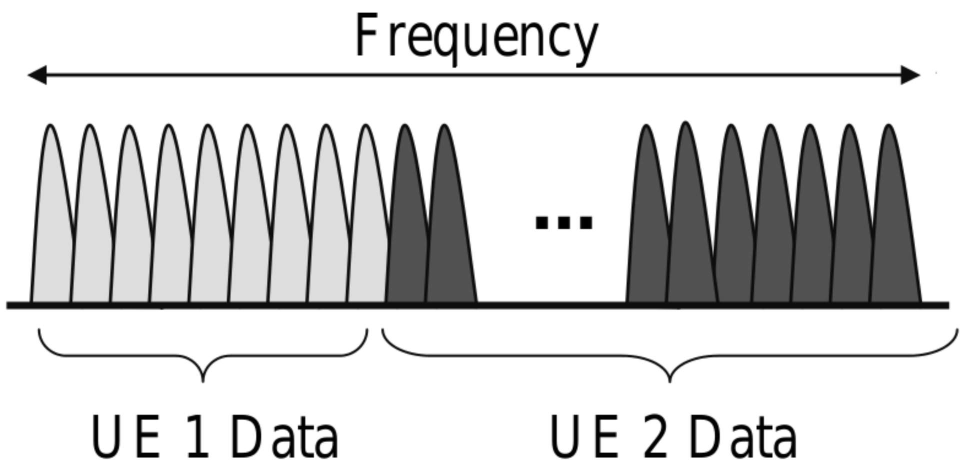


Figure 2-5: OFDMA

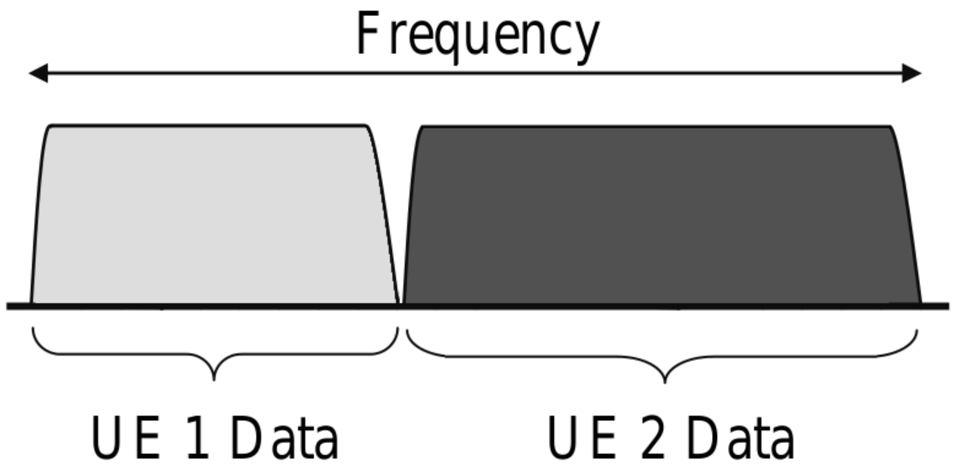


Figure 2-6: SC-FDMA

Both in downlink and in uplink, frequency resources are logically divided in Resource Blocks (RBs), each consisting of 12 subcarriers. The RB is the minimum allocation unit in the frequency domain, while in the time domain allocation is done with the resolution of the 1 ms subframe, defined within a frame of 10 ms (Figure 2-7).
The 1 ms subframe is referred as Transmission Time Interval (TTI). Note that the Physical Resource Block (PRB) in the specifications refers to the 0.5 ms slot. However, being resources allocated using the TTI resolution, in the following we

will always use the expression "RB" to refer to an amount of resources corresponding to 180kHz in the frequency domain and 1 ms in the time domain.
Each TTI the eNodeB scheduler allocates the RBs to the various UEs, selecting also the MIMO mode to use (see 2.1.5.2) and the MCS. For downlink, the MCS is chosen taking into account the Channel Quality Indicator (CQI) feedback specified by UEs. This CQI feedback can be either wideband or frequency selective. If wideband feedback is enabled, a single CQI value is specified by a UE for the whole system band. On the contrary, using frequency selective feedback a different CQI is specified for each *sub-band,* a sub-band being a group of contiguous RBs. Even if in the latter case the signaling overhead is bigger, selective feedback allows the scheduler to exploit multi-user diversity in the frequency domain, thus increasing system performance.

2.1.5.2 MIMO modes in LTE

LTE technology widely uses multi-antenna techniques in order to improve peak data rate, cell coverage and mean cell throughput. To achieve this various set of objectives, LTE adopts different MIMO technologies, including Spatial Multiplexing, Transmit Diversity and Multi-User MIMO [17].
The key idea of all MIMO techniques is to exploit also the spatial dimension of the wireless channel in order to get transmissions being either more efficient or more robust.

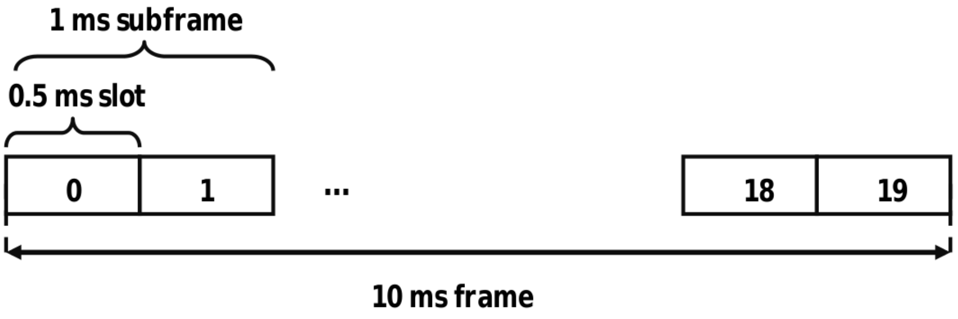


Figure 2-7: LTE frame structure

2.1.5.2.1 <u>Spatial Multiplexing</u>

Spatial Multiplexing allows the eNodeB to transmit independent data streams to a single UE on the same time-frequency resources exploiting different spatial channels, referred as *layers.* Using this technique the system bandwidth is virtually multiplied by a factor N, if N is the number of layers used.
There are two operation modes in Spatial Multiplexing: the closed- loop mode and the open-loop mode.
In the closed-loop mode, the eNodeB applies the spatial domain precoding on the transmitted signal taking into account the Precoding Matrix Indicator (PMI) reported by the UE, so that the transmitted signal matches with the spatial channel experienced by the UE. To support this mode in downlink, the UE needs to feedback the Rank Indicator (RI), the PMI and the CQI. The RI indicates the number of spatial layers that can be supported by the current channel experienced at the UE, while the CQI indicates the MCS the eNodeB should use to ensure that the block error probability will not exceed 10%.

Considering the case of M layers and N antennas, the precoding operation is defined by the relation

$$y = Wx$$

where x is the vector of M symbols in input to the precoder, W is the $N \times M$ precoding matrix and y is the vector of N symbols transmitted. The open-loop spatial multiplexing may be operated when reliable
PMI feedback is not available at the eNodeB (e.g. when the UE speed is not low enough or when the feedback overhead is too high). In this case, the feedback consists only of the RI and the CQI, and precoding operations are performed cyclically using a fixed set of precoding matrices.
Both in closed-loop and in open-loop Spatial Multiplexing, the maximum number of codewords a UE can receive is fixed to two, even if more spatial layers are available6. Therefore LTE defines the rules to map codewords to layers [5].
In LTE Release 8, Spatial Multiplexing is defined only for downlink and only for configurations with two or four transmit antennas. LTE Advanced extends the existing Spatial Multiplexing technologies, introducing the support for configurations with up to eight transmit antennas in downlink and up to four transmit antennas in uplink.

2.1.5.2.2 Transmit Diversity

Transmit Diversity is a MIMO transmission mode that exploits the spatial dimension of the wireless channel transmitting the same data stream over two or more different spatial layers, in order to increase the reliability of the transmission.
The Transmit Diversity scheme is specified in LTE Release 8 for configurations with two or four transmit antennas in downlink and with two transmit antennas in uplink. Both in downlink and in uplink there is always a single codeword for each UE.
In LTE Release 8 downlink, the Space-Frequency Block Code (SFBC) is used if the eNodeB has two transmit antennas, while for eNodeBs having four transmit antennas a combination of the SFBC and the Frequency-Switched Transmit Diversity (FSTD) is used to provide robustness against the correlation between channels from different transmit antennas and for easier UE receiver implementation. In LTE Advanced downlink, Transmit Diversity is defined also for eight transmit antennas.
In uplink, the transmit antenna selection for UEs with two transmit antennas is specified. The antenna to be used for transmission can be selected either by the eNodeB (closed-loop transmit antenna selection) or autonomously by the UE itself (open-loop transmit antenna selection). In LTE Release 8, SFBC is not employed in uplink to avoid the additional cost required to implement two power amplifiers at the UE. LTE Advanced instead introduces the use of uplink Transmit Diversity with up to four transmit antennas.

2.1.5.2.3 Multi-User MIMO

Multi-User MIMO (MU-MIMO hereafter) is a generalized form of Spatial Multiplexing which allows the eNodeB to allocate multiple UEs in the same time-frequency resources, thus adding more flexibility to scheduling operations. MU-MIMO does not increase peak data rate, as Spatial Multiplexing does, but makes it possible to increase cell throughput even in presence of legacy UEs, having a

single receive antenna, or when channel conditions are not suitable for Spatial Multiplexing.
This transmission mode is supported both in uplink and in downlink by the LTE standard.
In uplink, the eNodeB can always schedule more than one UE to transmit in the same time-frequency resources, thus forming a MU- MIMO configuration known as Virtual MIMO (VMIMO). However, in order to be able to correctly differentiate and demodulate the signals transmitted by these UEs, the eNodeB needs to assign orthogonal reference signals to them.
In downlink, if a UE is configured to be in MU-MIMO, only rank-1 transmissions can be scheduled to that UE: that is a single codeword on a single layer is transmitted. The eNodeB can schedule multiple UEs in the same time-frequency resources using different rank-1 precoding matrices. The UE receives only the information about its own precoding matrix and decodes the signals received using the common reference signal together with the precoding information obtained from the control signaling.
For MU-MIMO transmission mode, the UE generates the PMI/CQI feedback without any knowledge about other simultaneously scheduled UEs. Hence, there could be a mismatch between the UE’s CQI report and the actual CQI experienced, due to the lack of knowledge of the interference caused by another UE scheduled simultaneously.
In LTE Advanced further enhancements have been proposed for MU-MIMO in order to improve system throughput beyond what is achieved in LTE. It is worth mentioning here the use of a maximum of four spatial layers, with up to two layers per UE, and the possibility to use an enhanced Channel State Information (CSI) feedback, using a two-matrix framework [6].

3 PROPOSED SCHEDULER

3.1 Introduction

In this chapter we investigate resource allocation in the downlink of a MIMO-capable LTE-A system at the MAC layer. The purpose is to provide a scheduling framework which accommodates different types of traffic, *and* exploits radio resources efficiently. We first show that computing the *maximum throughput allocation* in a MIMO-enabled LTE cell is practically unfeasible given the number of UEs involved and the scheduling timescale. Hence, we propose to separate the resource allocation problem into two sub-problems, namely *transmission mode selection (TMS)* and *frequency-domain packet scheduling* (*FDPS*). For the TMS problem, we propose a tunable polynomial-time algorithm. For the FPDS sub-problem, we propose a modular framework that allows different types of services to be accommodated *on demand*. The basic block is an opportunistic allocation, which maximizes the throughput. Other blocks can be added if traffics with fairness (i.e., minimum throughput) or QoS (i.e., deadline) constraints are to be considered. We show via simulation that the opportunistic version of the FPDS achieves near-optimal cell throughput, and that adding QoS and fairness only entails a modest throughput loss.
To the best of our knowledge, no previous work covers multi-service scheduling under MIMO constraints. Those which consider single aspects (e.g., only throughput maximization, or fairness, under MU-MIMO) often rely on simplifications and idealizations that make them unsuitable for the LTE-A environment. Some either ignore MU-MIMO (e.g. 16), or assume arbitrary, out-of-standard MU-MIMO UE pairings in the same TTI (e.g. 17-18), others neglect the TMS problem (which greatly influences the cell throughput) by forcing *a priori* a single transmission mode (e.g. 19-20).

3.2 LTE-A Resource allocation and constraints

Hereafter we describe the LTE-A system, focusing on those features which are more relevant to the resource allocation problem in the downlink direction at the MAC layer.
In LTE, transmissions are arranged in time slots called Transmission Time Intervals, (TTIs), whose duration is 1ms. At the *logical* (MAC) level, the eNodeB allocates a vector of (*Virtual*) *Resource Blocks* (RBs) to its associated UEs on each TTI. Each RB carries a fixed number of symbols, which translate to different amounts of bits depending on the *modulation and coding scheme* (MCS) used by the eNodeB on that RB. In general, more information-dense modulations (e.g., up to 64QAM, which yields 6 bits per symbol) are favored when a better channel to the UE is perceived. The quality of the wireless channel, in fact, varies over both time and frequency. For this reason, UEs report their perceived downlink channel state to the eNodeB as a Channel Quality Indicator (CQI). The CQI is an index in a standard table, computed by the UE according to the measured Signal to Noise Ratio (SNR), and determines the MCS that the eNodeB should use and the number of bits per RB, called Transport Block Size, TBS. Accordingly, we will

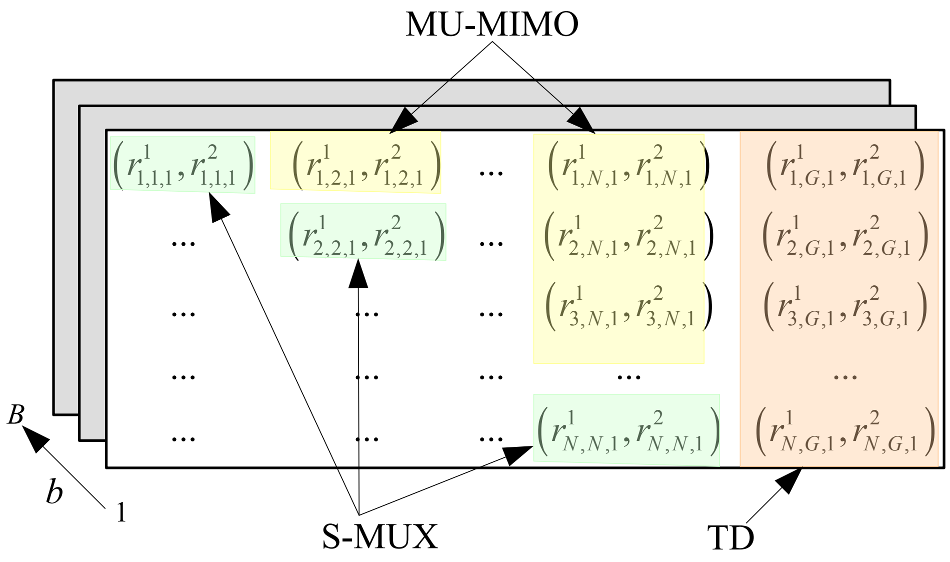
sometimes use the word CQI to refer either of the latter two, trading a little description accuracy for conciseness and ease of reading.
Depending on the settings, either one *wideband CQI* or several *per logical sub-band x CQIs* can be reported. In the first case, the CQI is an average measure of the channel quality over the whole OFDMA frequency spectrum. In the latter case, the OFDMA spectrum is partitioned in *logical sub-bands (LSBs)*, and UEs report the average channel status on each of these. In the latter case, an increased reporting overhead is the price to be paid to enable the eNodeB to exploit *frequency selectivity*, i.e. to schedule UEs on those RBs where a higher-order MCS can be exploited. While the eNodeB is free to use whichever MCS it sees fit (regardless of the reported CQI) to address a UE, exceeding the reported CQI increases the likelihood of decoding errors at the UE. Retransmissions are handled by the *hybrid-ARQ (H-ARQ)* processes, which eat out some RBs from subsequent TTI frames. While a certain amount of retransmissions is unavoidable in a wireless environment, it is obvious that a well-designed scheduler should not waste resources by pushing UE receivers beyond their limits.
In order to build a frame in a TTI, the eNodeB scheduler selects *which UEs* are going to be targeted, using *which transmission mode* (e.g., S-MUX, TD, MU-MIMO), in *which LSBs* (in case of frequency-selective CQIs) using *which MCS* to guarantee reliable transmission. These decisions are made, either sequentially or jointly, based on the reported CQIs, the backlog and type of traffic of each UE, the QoS requirements, etc. A scheduler might aim to maximize the cell throughput, to ensure fairness among UEs (i.e., to guarantee that UEs reporting low CQIs for many TTIs do not starve), or to meet deadline guarantees with real-time traffic such as VoIP, or to combine any of the above features.
The key idea of all MIMO techniques is to exploit the spatial dimension of the wireless channel in order to achieve more efficient or robust transmissions. At the MAC layer, MIMO transmission implies sending *twice* the usual amount of information on one RB, as if *space* were a third dimension. MIMO modes differ as for i) whether one or two different *codewords* are transmitted, and ii) whether one or two *UEs* are targeted simultaneously. This yields a total of three combinations: one codeword to one UE, however replicated to increase likelihood of reception (Transmission Diversity), two codewords to the same UE (Spatial Multiplexing), two codewords to two UEs (Multi-User MIMO).
The practicability of each transmission mode for a given UE depends on information fed back by the UEs, such as the Precoding Matrix Indicator and Rank Indicator. Depending on the latter two, in fact, a UE may or may not support more demanding modes, such as S-MUX. Furthermore two UEs may or may not be paired MU-MIMO depending on these. While these aspects are indeed crucial to the LTE system, our scheduling framework is largely independent of physical details. For this reason, we refer the interested reader to the standards [2] or to the abundant literature on the subject (e.g., [8]).
A UE reports one CQI per codeword, i.e. one per MU-MIMO and TD, and two for S-MUX. The CQI is related to the MIMO mode the UE is addressed with. This means that the CQI of a UE served in TD may not describe the channel quality related to a hypothetical service in S-MUX or MU-MIMO in the same TTI. This makes the TMS problem all but a wild guess, unless correlation between the channel reports of the various MIMO modes is assumed. Such correlation has been analyzed in [25], where authors – using detailed link-level simulations – show that the CQI for one

MIMO mode can be translated to that of another one by applying a suitable offset. For instance, if a UE reports a CQI equal to *c* for both codewords in S-MUX, it is likely that it can be served with a higher CQI if paired in MU-MIMO with another UEs (under the physical conditions that allow so), as the channels to two *different* UEs would be *more* spatially separated than those to the same UE.
There are some constraints that are inherent to resource allocation in LTE-A. Although they do complicate the problem, neglecting them leads to unfeasible allocations. The first constraint will be called *one-MIMO-mode*: a UE can only use *one* of the above MIMO modes in a TTI. For instance, it cannot receive data using TD on one RB and S-MUX on another RB. The second constraint further specifies the first one for MU-MIMO allocations, and will be called *MU-MIMO-pairing* constraint: UE i must be paired with only one UE j in MU-MIMO in a TTI, and both UEs have to be allocated on exactly the same RBs. The above constraints are related to the TMS decision. The third one is instead related to the MCS selection, and will be called *one-MCS* constraint: a codeword must be transmitted to a UE using the *same MCS* on all the RBs it occupies, despite the fact that the UE may report different CQIs on these RBs (e.g., when these belong to different LSBs). In order not to increase the error rate, the minimum CQI should be adhered to for all the RBs in a codeword. Hence, the amount of data that an UE can reliably receive is, in general, less than the sum of the TBS corresponding to the CQIs reported for each RB.

3.3 System Model and Throughput-Optimal Scheduling Problem

We focus on the downlink of an LTE-A cell, which is managed by a the eNodeB downlink MAC scheduler. The scheduler serves N UEs. A UE may have one or more streams of traffic queued at the eNodeB. We denote with T_i the overall backlog of UE i, which we assume to be arbitrary divisible. During each TTI, the scheduler allocates a frame of M RB to the UEs. We assume that the RB are grouped into B logical sub-bands (LSB), and M_b is the number of RBs on LSB b, $1 \le b \le B$. We assume that UEs can use TD, S-MUX and MU-MIMO, and that the scheduler can alternate among the three modes on a per-TTI basis. Furthermore, we assume that the eNodeB can multiplex at most two codewords on the same RB, i.e. up to two streams can be sent, whether to the same UE (S-MUX) or to a pair (MU-MIMO).
We assume that *full feedback* is available at the eNodeB, meaning that the latter is aware of the rate that it may use on every LSB to address a UE (or pair thereof) for each MIMO mode. The means through which such information are made available at the scheduler concern the physical layer, and are thus outside the scope of this paper. To remark the viability of this hypothesis, however, we recall that it is customary to compute estimates of the rates in the various MIMO modes by offsetting standard UE reporting 25.
The above information is summarized at the eNodeB in a three-dimensional *rate matrix* (RM), shown in Figure 3-1. The latter has N rows, $N+1$ columns, and B layers in the third dimension. Each element of the matrix is a couple $\left(r_{i,j,b}^{1}, r_{i,j,b}^{2}\right)$.

Value $r^1_{i,j,b}$ (resp. $r^2_{i,j,b}$), with $i \neq j,\ j \neq n+1$, represents the rate (i.e. the number of bits) that can be achieved by UE *i* (resp. UE *j*) when UEs i, j are scheduled together in MU-MIMO on a RB of LSB b. When $i = j$, the same information represents the S-MUX rate for either stream of UE *i*. Finally, when $j = N+1$, we assume that $r^1_{i,N+1,b}$ is the rate for UE *i* served in TD, and we assume that $r^2_{i,N+1,b} = 0$ $\forall i, b$. We will sometimes use the sum $m_{i,j,b} = r^1_{i,j,b} + r^2_{i,j,b}$ as the *sum rate* of the matrix element. Since, under MU-MIMO (i.e. when $i \neq j,\ j \neq n+1$), it is $r^1_{i,j,b} = r^2_{j,i,b}$, we can limit ourselves to considering only the matrix elements with $j \geq i$. For ease of exposition, we will sometimes refer to "pairs of UEs" served on an RB, overlooking the fact that a single UE is targeted in S-MUX (in which case it is "paired with itself") and in TD (in which case it is paired with the *ghost* UE $G = N+1$). If needed, the present model allows for a single, wideband CQI (in which case $B = 1$). We leave it to the interested reader to simplify the algorithms presented in the following under that condition. We also observe that the interference from neighboring eNodeBs can be incorporated in this model, just by reducing the UE's rates on certain LSBs.



MU-MIMO
$(r^1_{1,1,1}, r^2_{1,1,1})$
$(r^1_{1,2,1}, r^2_{1,2,1})$
$(r^1_{1,N,1}, r^2_{1,N,1})$
$(r^1_{1,G,1}, r^2_{1,G,1})$
$(r^1_{2,2,1}, r^2_{2,2,1})$
$(r^1_{2,N,1}, r^2_{2,N,1})$
$(r^1_{2,G,1}, r^2_{2,G,1})$
$(r^1_{3,N,1}, r^2_{3,N,1})$
$(r^1_{3,G,1}, r^2_{3,G,1})$
$(r^1_{N,N,1}, r^2_{N,N,1})$
$(r^1_{N,G,1}, r^2_{N,G,1})$
B
b
1
S-MUX
TD

Figure 3-1: the Rate Matrix

CQI	No. of bits
0	0
1	2
2	3
3	5
4	11
5	15
6	20
7	25
8	36
9	38
10	49
11	63
12	72
13	79
14	89
15	92

Table 3-1: Number of bits per CQI

We assume for simplicity that the number of bits that can be transmitted to the UE is *linearly increasing* with the number of RBs allocated to it. That number is determined by the CQI reported in Table 3-1, which is obtained by averaging the values reported in [5].

Our goal is to devise a resource allocation algorithm that i) accommodates different types of traffic simultaneously; and ii) exploits frequency diversity among the LSBs, thus utilizing the radio resources effectively.

Before explaining our solution (which is the subject of the next section), we show that maximizing resource effectiveness, i.e., objective iii) alone, under the LTE constraints (i.e., one-MIMO-mode, MU-MIMO-pairing, one-MCS) is beyond achievable in practical scenarios. We formulate the *maximum-throughput allocation problem* (MTAP) as an optimization problem, and show that solving it optimally takes an unfeasible time.

The MTAP is reported in Figure 3-2, under a *full buffer* approximation[1] (i.e., $\forall i, T_i = \infty$). Binary variables $c_{i,j}$ state whether i,j are paired or not, while $x_{i,j,b}$ state whether i,j will be allocated RBs in LSB b, the number of those RBs being represented by $A_{i,j,b}$. Each UE *i* has two *rate variables* ρ_i^1, ρ_i^2, representing the UE's rate on the two streams it may receive. The objective is to maximize the overall rate on all the LSBs. Constraint *(i-ii)* model the *one-MIMO-mode* and *MU-MIMO-pairing* constraints. Constraints (*iii)* states that a pair can only be allocated RBs out of a given LSB if they are paired at all. Constraints *(iv-v)* ensure that no more than M_b RBs are allocated on each LSB, and that $A_{i,j,b} > 0$ if $x_{i,j,b} = 1$. Constraint *(vii)* implements the *one-MCS* constraint. Let Y be a very large positive constant: for each LSBs where (i,j) actually have RBs allocated (i.e., those with $x_{i,j,b} = 1$, otherwise the constraints are obviously verified), the rate of the codewords cannot exceed the achievable rates. Note that, since ρ_i^1, ρ_i^2 appear only in those constraints, as well as in the objective function with a positive sign, some constraints of this block will be *active* at the optimum, i.e. equality will hold in some of *(vii)*, which implies that the rate used on each codeword will actually be one of those reported in the RM. Constraint *(viii-ix)* express that $c_{i,j}$, $x_{i,j,b}$ are binary, and that ρ_i^1, ρ_i^2, $A_{i,j,b}$ are integer.

[1] *Taking into account finite buffers in the MTAP is not conceptually difficult, but requires one to model* padding*, i.e. the number of bits that do not contain useful information in a RB. This makes the model more cumbersome than it already is.*

$$
\begin{array}{lll}
\max \sum_{b=1}^{B} \sum_{i=1}^{N} \sum_{j=i}^{G} \left(\rho_i^1 + \rho_j^2\right) \cdot A_{i,j,b} & & \\
s.t. & & \\
\sum_{j=i}^{G} c_{i,j} = 1 & \forall i & (i) \\
\sum_{i=1}^{j} c_{i,j} = 1 & \forall j, j \neq G & (ii) \\
x_{i,j,b} \leq c_{i,j} & \forall i,j,b & (iii) \\
A_{i,j,b} \geq x_{i,j,b} & \forall i,j,b & (iv) \\
A_{i,j,b} \leq x_{i,j,b} \cdot M_b & \forall i,j,b & (v) \\
\sum_{i=1}^{N} \sum_{j=i}^{G} A_{i,j,b} \leq M_b & \forall i,j & (vi) \\
\rho_i^v \leq r_{i,j,b}^v + Y \cdot \left(1 - x_{i,j,b}\right) & \forall i,j,b,v & (vii) \\
c_{i,j} \in [0,1] \, \forall i,j, \quad x_{i,j,b} \in [0,1] \, \forall i,j,b, & & (viii) \\
\rho_i^v \in Z^+ \, \forall i,v, \quad A_{i,j,b} \in Z^+ \, \forall i,j,b & & (ix)
\end{array}
$$

Figure 3-2: Mathematical model for the MTAP

The MTAP has $O\left(N^2 B\right)$ variables, a quadratic objective function and $O\left(N^2 B\right)$ linear constraints. This makes it an integer quadratic optimization problem. Unfortunately, the objective function is non-convex, which makes the problem particularly hard to solve even at small scales (non-convex problems are NP-hard in general). The non-convexity follows from the *one-MCS* constraint: without the latter, ρ_i^1, ρ_i^2 variables could be replaced by constants $r_{i,j,b}^1, r_{i,j,b}^2$, which would make the objective function linear. Broadly speaking, selecting *both* an arbitrary number of LSBs *and* a single transmission rate as a function of per-LSB reports inevitably leads to a quadratic (hence non-convex) objective function. For instance, using the *average* CQI (instead of the minimum) would not make a difference.
One might wonder whether computing a resource allocation neglecting the *one-MCS* constraint, and *then* superimposing it in order to make the allocation feasible, could be a viable option. On one hand, this would still involve solving an integer-linear problem (which is probably too hard to be solved in a millisecond). On the other hand, it may lead to largely suboptimal throughputs, as shown in the following example.

Example:
Consider a system with 2 UEs and 2 LSB, each one with only one RB. Assume the following RM:

$$
RM_1 = \begin{bmatrix} (10,10) & (9,9) & (11,0) \\ - & (2,2) & (3,0) \end{bmatrix} RM_2 = \begin{bmatrix} (5,5) & (9,9) & (6,0) \\ - & (3,3) & (4,0) \end{bmatrix}
$$

The maximum throughput is achieved with both UEs in MU-MIMO, and it is equal to $(9+9)\cdot 2 = 36$. Without the *one-MCS* constraint, instead, the throughput-optimal allocation is achieved with both UEs in S-MUX, and its throughput is:

$$(10+10)+(5+5)+(2+2)+(3+3)=40$$

However, if the *one-MCS* constraint is enforced to make the latter allocation feasible, the overall throughput is reduced to:

$$2\cdot(\min\{10,5\}+\min\{10,5\})+2\cdot(\min\{2,3\}+\min\{2,3\})=28,$$

i.e. considerably less than the optimum.

3.4 Polynomial-time Opportunistic Scheduling

In order to make the problem tractable, we split it in two: first we solve the TMS problem, and then we solve the allocation problem given a TMS, as in

Figure 3-3. The TMS algorithm takes the RM as an input, and produces a *pair list L* as an output. The *allocation function* takes the pair list as an input, and actually *allocates* the RBs to the UE pairs, i.e., outputs an *allocation list* of up to M tuples $\{i,j,b\}$, which actually populate the frame for the TTI.

Such partition comes with a twofold advantage. First of all, it allows us to partition the LTE constraints as well, thus greatly simplifying the problem: the *one-MIMO-mode* and *MU-MIMO-pairing* constraints are taken care of in the TMS algorithm, and the *one-MCS* constraint is instead considered in the allocation function. Second, it allows different algorithms to be plugged in each block, e.g. to achieve different trade-offs between complexity and effectiveness. For instance, different allocation functions can be envisaged to account for fairness, QoS, throughput effectiveness, and so on, without modifying the TMS algorithms. We will actually exploit this concept in section 3.5.

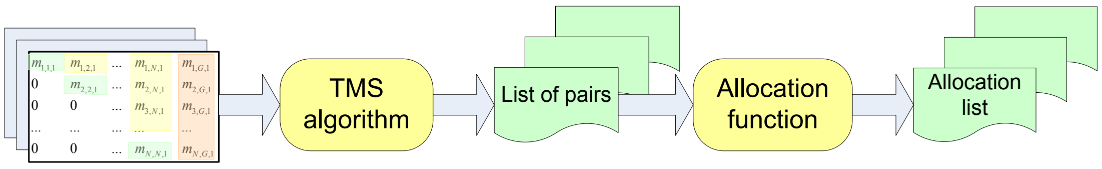

$m_{1,1,1}$ $m_{1,2,1}$ ... $m_{1,N,1}$ $m_{1,G,1}$
0 $m_{2,2,1}$ ... $m_{2,N,1}$ $m_{2,G,1}$
0 0 ... $m_{3,N,1}$ $m_{3,G,1}$
0 0 ... $m_{N,N,1}$ $m_{N,G,1}$
TMS algorithm
List of pairs
Allocation function
Allocation list

Figure 3-3: The resource allocation framework

3.4.1 TMS algorithms

The objective of a TMS algorithm is to favor, for each UE, the transmission modes that *might* lead to a higher throughput. "*Might*", since the throughput will depend on which LSBs are allocated to which UE pair, a decision which is made by the allocation function only later. One should decide, at this stage, whether it is preferable to select a transmission mode that exhibits a high rate peak on possibly a single LSB and a poor rate everywhere else, or one whose peak is less pronounced, but consistent on more than one LSB. We propose an algorithm called *K-max*, which can be tuned to trade off between frequency selectivity and average throughput, thus allowing either of the extreme behaviors in the above example.

K-max computes a bi-dimensional $N \times G$ *score matrix* (SM), whose elements $s_{i,j}$ are the sum of the *K* largest sum rates $m_{i,j,b}$ over the third (*b*) dimension in the RM. *K* is a tunable parameter. When $K = 1$, the algorithm selects $s_{i,j} = \max_{1 \le b \le B}\{m_{i,j,b}\}$, i.e. the *largest* sum rate achieved by pair (i,j) on the whole frame. For $K = B$, it selects $s_{i,j} = \sum_{b=1}^{B} m_{i,j,b}$, i.e. the sum of the sum rates on all the LSBs. Practically speaking, the value of K determines the number of LSBs deemed meaningful to contribute to rate peaks.

The *K-max* explores the SM cyclically as follows:

- select $(i,j) = \underset{1 \le i \le N, 1 \le j \le G}{\arg\max}\{SM\}$ from the SM
- add (i,j) to the pair list *L*;
- purge the SM for the next iteration: zero the i^{th} row and column and, if $i \ne j \wedge j \ne G$ (i.e., MU-MIMO is being considered), zero the j^{th} row and column as well (consistent with the *one-MIMO-mode* and the *MU-MIMO-pairing* constraints).

The cycle terminates when all UE pairs have been assigned a transmission mode. The pseudo-code of *K-max* is shown in Figure 3-4.

K-max algorithm

```
foreach pair (i,j) in RM
|     SumK(i,j)=0
|     for k=1 to K, Largest[k]=0
|     for b=1 to B
|     |     if ($m_{i,j,b}$ > Largest[K])
|     |     |     insert $m_{i,j,b}$ in the heap
|     |     |     remove Largest[K]
|     |     |     SumK(i,j)= SumK(i,j)+$m_{i,j,b}$-Largest[K]
|     |     endif
|     endfor
endfor
repeat
|     select (i,j)=argmax(SM)
|     add (i,j) to pair list L
|     for h=i to G, SM(i,h)=0
|     for k=1 to i-1, SM(k,i)=0
|     if (i!=j and j!=G)
|     |     for h=j to G, SM(j,h)=0
|     |     for k=1 to j-1, SM(k,j)=0
|     endif
until no non-null elements are left in SM
```

Figure 3-4: K-max algorithm pseudo-code

In the description of the *K-max* algorithm reported in Figure 3-4 we favored clarity over complexity. Under a slightly more complicated (but still intuitive enough) implementation, the complexity of the *K-max* algorithm is the following:

Property 1: *the K-max algorithm can be implemented at a cost equal to* $O\left(N^2\left(\log N + B\log K\right)\right) \approx O\left(N^2 \log N\right)$.

Proof: Computing the SM from the RM requires at most B insertions or replacements of a set of K elements. Hence $O\left(B\log K\right)$ operations per pair $\left(i,j\right)$ are required, i.e. it is $O\left(N^2\left(B\log K\right)\right)$. Sorting the SM elements in a max-heap takes $O\left(N^2 \log N\right)$ operations. Every iteration in the TMS selection cycle requires:

- selecting the top element from the max-heap
- locating the $O\left(N\right)$ elements on the row(s)/column(s) to be canceled and extracting them from the max-heap

If the elements of the same row/column are linked together in any order via pointers (which comes with no additional overhead) the elements to be extracted from the max-heap can be accessed directly (i.e. without navigating the entire heap). Thus, the $O\left(N\right)$ extractions and rehapifications required for an iteration have a cost of $O\left(N\log N\right)$. Since the number of pairs to be formed cannot exceed N (this limit is in fact achieved when no MU-MIMO pairs are formed), at most $O\left(N\right)$ iterations of the selection cycle are required, which makes the *K-max*

algorithm $O\left(N^2 \log N\right)$. Summing up the two contributions, and observing that K and B are small constants, the property follows.
The rationale of the *K-max* algorithm is to allow only the *best* values to contribute to the score of a pair (i,j). The same objective can be achieved through a slightly different algorithm, which we call *shift-and-sum*. The latter only differs from the *K-max* in that the SM is computed as follows:

$$s_{i,j} = \sum_{b=1}^{B} \left\lfloor \frac{m_{i,j,b}}{2^k} \right\rfloor.$$

This way only "large" $m_{i,j,b}$ values are going to contribute to the SM entries, small values being zeroed by the floor operator. The name *shift-and-sum* is due to the fact that the integer division by 2^k is a right shift operation. The alert reader will observe that the complexity is the same as the previous one. Both the *K-max* and the *shift-and-sum* depend on a tunable parameter.
Finally, we observe that the TMS algorithm does not take into account the finiteness of the UE buffers. More specifically, a UE *i* might not have enough traffic to exploit K peak LSBs, and still be selected by the TMS ahead of another having a heavier backlog and a worse channel. While finite backlogs are taken care of by the allocation function, a straightforward optimization is to restrict the TMS to the set of *backlogged* UEs, which is what we give for granted from now on.

3.4.2 Allocation function

We now detail how to build the allocation list from the pair list. In doing so, the *one-MCS-mode* constraint is enforced, and finite backlogs for UE queues at the eNodeB are also taken into account. We first present an allocation function that aims at maximizing the throughput, and then (in the next subsection) show how to enhance it to consider QoS and fairness.
The problem with the *one-MCS-mode* constraint is that, as already explained, the overall rate for a UE pair is given by the *minimum* TBS, times the number of allocated RBs. At some point, allocating one more RB might actually *decrease* the overall throughput, if the MCS on the new RB is smaller than the minimum achieved thus far, and this should obviously be prevented. In order to steer the allocation towards the maximum throughput, we dynamically compute a *gain* associated to each pair, and allocate RBs to the one that exhibits the maximum gain. The gain depends on the pair (i,j), on the LSB b, on the buffers T_i and T_j (this last if $j \neq G$), on the achievable rates on already allocated RBs, and represents the actual *increment* in the number of bytes that would be transmitted if that RB was given to that pair, under the *one-MCS mode* constraint.
The pseudo-code for the *max-gain* algorithm is reported in Figure 3-5. In short, it is a cycle through all the RBs, at each iteration of which the RB with the maximum gain is selected and appended to the allocation list. With reference to the initialization function, we store the number of usable RB for each LSB in the FreeRB vector, the RBs allocated to each pair in the AllocRB vector, and the residual backlog in each UE buffer through the ResBklog vector. In this last case, it is formally convenient to associate a null backlog to the *ghost* UE *G*. The rates of

either stream for all the UEs are set to the maximum achievable rate. Because of the *one-MCS-mode* constraint, it can actually happen that the overall throughput varies non-monotonically through the iterations. To cope with this, we record the maximum throughput achieved so far and the iteration at which it is attained. The CurTP, MaxTP and MaxTPPointer serve this very purpose.

With reference to Figure 3-5, the gain for MU-MIMO pair (h,k) in a RB of LSB a cannot exceed the backlog on the two UEs, and may require the rate of either or both streams to be decreased to the one supported on LSB a. For a TD user, the second (*ghost*) stream always has a null rate and backlog, hence the same computation still holds. The computation for the S-MUX case is slightly different, as both streams draw traffic from the same buffer.

The allocation function consists of a cycle, repeated until there is nothing more to transmit or all the blocks have been allocated. Pairs are removed from the pair list once the residual backlog for both UEs is null, and depleted LSBs are not considered. If only *one* UE in the pair is backlogged, the other gets padding bits (which do not contribute to the gain, in any case). In the allocation cycle, the gain is computed for all the relevant combinations of pairs and LSBs, and the combination that yields the maximum gain (whether positive or negative) is added to the list. Backlogs and free RBs in the LSBs are updated accordingly, and – to meet the *one-MCS-mode* constraint, the rate of each stream is set to the minimum rate achievable on the RBs allocated so far. The function keeps track of the system throughput, so that at the end the allocation is enforced up to the point where the throughput is maximum.

Initialization

```
foreach LSB b FreeRB(b)=$M_b$
foreach UE i ResBklog(i)=$T_i$
foreach (h,k) in the pair list L
|       $\rho_h^1$=MaxRate
|       $\rho_k^2$=MaxRate
|       AllocRB(h,k)=0
ResBklog(G)=0
CurTP=0
MaxTP=0
MaxTPPointer=nil
```

Compute gain function

```
In parameters: pair (h,k), LSB a
if (h!=k)              // MU-MIMO or TD
|     gain= min(  (AllocRB(h,k)+1)*min($\rho_h^1$, $r_{h,k,a}^1$),ResBklog(h) )+
            min(  (AllocRB(h,k)+1)*min($\rho_k^2$, $r_{h,k,a}^2$),
             ResBklog(k)) -   AllocRB(h,k))*($\rho_h^1+\rho_k^2$)
else                        // S-MUX
|     gain=min(   (AllocRB(h,h)+1)*(min($\rho_h^1$, $r_{h,h,a}^1$)+min($\rho_h^2$, $r_{h,h,a}^2$))
,ResBklog(h))-AllocRB(h,k))*($\rho_h^1+\rho_k^2$)
endif
```

Allocation Function

```
repeat
|     foreach LSB b such that FreeRB(b)>0
|     |     foreach (i,j) in the pair list L
|     |     |     compute gain(i,j,b)
|     let mg=gain(h,k,a) be the maximum gain
|     CurTP=CurTP+mg
|     append item {h,k,a} to the alloc. list
|     if (CurTP>MaxTP)
|     |     MaxTPPointer = this item
|     |     MaxTP=CurTP
|     FreeRB(a)=FreeRB(a)-1
|     AllocRB(h,k)=AllocRB(h,k)+1
|     $\rho_h^1$=min($\rho_h^1$, $r_{h,k,a}^1$)
|     $\rho_k^2$=min($\rho_k^2$, $r_{h,k,a}^2$)
|     ResBklog(h)=max(0,Th-AllocRB(h,k)*$\rho_h^1$)
|     ResBklog(k)=max(0,Tk-AllocRB(h,k)*$\rho_k^2$)
|     if(max(ResBklog(h),ResBklog(k))==0)
|     |     remove (h,k) from the pair list L
until L is empty or max(FreeRB(b))=0
allocate all RBs in the alloc. list up to MaxTPPointer
included
```

Figure 3-5: Pseudo-code of MaxGain algorithm

The allocation function lends itself to some obvious optimizations, which have been withheld for ease of reading: for instance, after an RB from LSB b is allocated to pair (i,j), the gain values for other pairs need not be computed anew, which saves a considerable overhead. Rather counter intuitively, instead, it is wrong to stop the allocation cycle when the maximum gain is non-positive, as shown in the following toy example. Assume that there are three LSBs, each with one RB, and one UE in

TD mode with an infinite backlog. Let 10, 4, 4 be the rates achievable by the UE on each RBs. The allocation cycle will select the maximum gain 10 in the first iteration. At the next iteration, however, the gain will be equal to -2 for either of the remaining two RBs. If the cycle is aborted, we end up with a throughput of 10. However, the obvious optimum solution is 12, achieved when *all* the RBs are allocated. The example also confirms that the system throughput is not necessarily monotonic under the *max-gain* algorithm.
As for complexity, we can state the following:
Property 2: *the max-gain algorithm can be implemented at a cost equal to* $O(N \cdot M \cdot B)$.

Proof: The allocation cycle is repeated $O(M)$ times. In the latter, $O(B)$ gains are computed (at a constant cost) for all the $O(N)$ backlogged pairs. The thesis follows straightforwardly.

3.5 QoS and Fairness Constraints

The resource allocation framework proposed in the previous section can be enhanced to accommodate deadline-constrained traffic and to avoid starvation of UEs with persistently low CQIs – i.e., to enforce a basic inter-UE fairness. This can be done by modifying the *allocation function*, without touching the TMS, as shown in Figure 3-6. More specifically, the allocation cycle described in the previous section is split into three cycles:

1. a first step (*QoS step*) where UEs with *urgent data*, i.e. with a backlog whose deadline is about to expire, are allocated RBs;
2. a second one (*fairness step*) where UEs with *excellent relative rate* conditions are allocated RBs;
3. and a last one (*efficiency step*), where RBs are allocated according to the highest gain, as shown in the previous section.

The first two steps can be configured to allocate a maximum amount of RBs, hence fine-tuning the trade-off between throughput efficiency, on one side, and QoS guarantees and/or fairness, on the other.

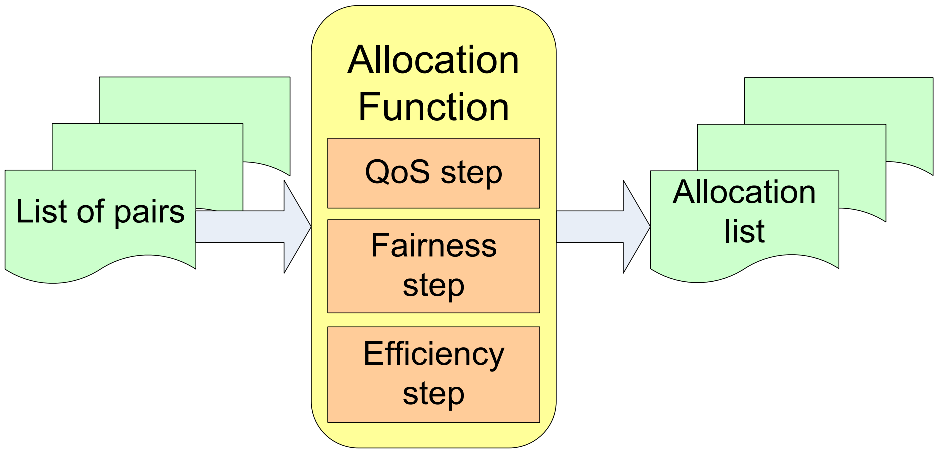


Figure 3-6: The enhanced allocation function

As far as QoS is concerned, we allow for real-time traffics of up to P different classes, sorted by decreasing priorities. For each class $p = 1..P$, an *Urgent Queue* (UQ)[34][35][37] U_p stores UEs references in FIFO order. To each class, a *deadline* D_p and a *slack time* S_p are associated, with $S_p < D_p$. The latter marks the threshold after which a packet starts to be considered *urgent*. When the deadline is less than S TTIs ahead, the UE the packet belongs to is inserted into the related UQ. The UE is removed from the UQ when all its urgent data has been served. A *maximum burst* M_p can also be configured to cap the amount of data that a single UE (having urgent data) can transmit in a TTI, hence preventing it from hogging all the available RBs.

3.5.1 QoS Step

The QoS step consists of two functions: a *tagger*, to be run at the beginning of a TTI, which inserts UEs having urgent data into UQs. Another function *selector* visits the UQs according to their priority and actually allocates RBs to the UEs. To maximize the efficiency, RBs are allocated on the LSBs where the highest CQIs are registered. If a urgent UE *i* is paired in MU-MIMO with another UE *j* (either non-urgent, or having a lower priority), the latter will also enjoy some (undue) extra service in he QoS step. In this last case, care is taken so that the latter is not serviced *twice* during the step.
To make the scheduling framework consistent, UEs that are served from the UQs in the QoS step should be *marked* so that subsequent allocations (whether in the fairness or in the efficiency step) do not jeopardizes the transmission of urgent data. This may happen, albeit infrequently in practice, if a UE is scheduled later on in a different LSB, and its MCS decreases because of the *one-MCS* constraint. In this case, they could end up transmitting *fewer* data than allocated in the QoS step. To prevent this from happening, we store the allocated data of UEs dequeud from UQs in a variable UrgBklg, and modify the compute gain function (Figure 3-5) so that a gain equal to $-\infty$ is returned should an allocation decrease the allocated data below UrgBklg.

```
tagger function
Let B be the list of user buffers
for all b ∈ B do
|     Let p be the class of service of b
|     Let h be the user owning b
|     Let d be the instantanous delay of the HOL packet of b
|     if $d > D_p$ and h is not already in $U_p$ then
|     |     Add h to $U_p$
|     end if
end for
end function
```

Figure 3-7: Tagger function pseudo code

```
selector function
Let U_p be the list of U_ps sorted in descending order of priority
Let tti be the current TTI
for all u_p ∈ U_p do
|     for all h ∈ u_p do
|     |     Let b_h be the buffer belonging to user h
|     |     Let S_h be the bytes already served from b_h this TTI
|     |     Let L_h be the list of user h's ∈ List of Pairs
|     |     Remove h from u_p
|     |     urgent_bytes = compute_urgent_bytes(b_h,tti)
|     |     if S_h >= urgent_bytes then
|     |     |     continue
|     |     end if
|     |     for all (LSB,h,k) ∈ L_h do
|     |     |     lte_allocation_function(LSB, h, k)
|     |     |     if h is no more urgent then
|     |     |     |     break
|     |     |     end if
|     |     end for
|     |     if h has still urgent bytes then
|     |     |     Put h in u_p again
|     |     end if
|     end for
end for
end function
```

Figure 3-8: Selector function pseudo code

The procedure *compute_urgent_bytes(buf, tti)*, given a user's buffer and the current TTI, computes the amount of urgent bytes in the buffer. Urgent packets are those that have been waiting in the buffer for *more* than the *deadline* D_p defined for the corresponding class of service *p*: the *youngest* urgent packet is that packet being in the buffer for $(D_p + 1)TTI$. Therefore, to compute the total amount of urgent bytes, the procedure sums the size of all packets whose arrival time is in the interval [hol_arrival_time, curren_tti $- D_p$). Let this number be *T*. Actually the procedure does not return necessarily the above computed *T*, because this number could be bigger than the *Max Urgent Burst* M_p defined for the class of service *p*. So the minimum between M_p and T is returned. Lte_allocation_function is an allocation procedure that follows the same rules described in Figure 3-5 while preserving the *one-MIMO-mode, MU-MIMO-pairing* and *one-MCS* constraint.

3.5.2 Fairness Step

As far as *fairness* is concerned, we reserve a configurable number of RBs (e.g., one per UE pair) to serving UEs whose channel conditions are good if compared to the recent history. This allows border-cell UEs to increase their chance to be served, thus avoiding starvation.

Fairness step works as follows. We keep track of the channel quality of each user over a time window of k TTIs. When a user has a local peak in his historical channel quality, that user is served on LSB where the peak is reached. To identify whether a user has a local peak or not we consider a threshold t $(t < k)$ beyond which the user's current channel quality is considered to be part of a peak. More precisely, if the current achievable rate is bigger than *t* of the *k* achievable rates in the window, we assume that the user has a local peak. In this situation the user is served on the LSB where the peak takes place.

Both *k* and *t* parameters are tunable and each class of service can have its own values. The *k* parameter influences the amount of time we want to take into account to estimate user's relative quality. The *t* parameter, instead, allows one to move the scheduler behavior between the two endpoints of fairness and opportunism. In fact, if *t* is small with respect to *k*, the probability of service is higher for each user at the expense of system efficiency; on the other hand, using values of *t* close to *k* we reduce the probability of service, but we are more selective in the identification of channel peaks, thus using higher MCSs on the average.

At the end of *Fairness* step historical channel information is updated for each user for whom we have new feedback data, regardless the fact that the user has been scheduled or not this TTI.

Historical channel information To keep track of this historical infor- mation about channel quality, for each user we keep a *UserHistory* structure. This data structure contains a circular queue, used to imple- ment the sliding window behavior. When a new achievable rate value is available, it is inserted in the queue, causing the drop of the oldest value stored.

This structure offers two utility functions helping to implement the above proposed algorithm:

- *push(UH, rate)* - This function is used to insert a new *rate* value in the window contained in *UserHistory UH.*
- *check(UH, rate)* - This function checks whether the given *rate* is bigger than t older values of *UserHistory UH,* returning a boolean value. More formally, the *check()* procedure returns *true* if the following condition is verified:

$$\sum_{i=1}^{k} 1_{(rate > r_i)} \geq t$$

  where r_i indicates the i-th historical channel quality value stored in the window and k is the window size.

Being in a system where frequency selectivity is considered and differ- ent transmission modes are available, the problem of *what* achievable rate should be recorded in the window arises. In our design we pro- pose to put into *UserHistory*

the achievable rate corresponding to the transmission mode selected by the TMS algorithm for the current TTI, considered on the RB where the user has the best quality.
After the (possible) allocation, we break the cycle over Li: for each user *Fairness* step serves at most one RB, that is the RB where the user has a local peak. Before considering another user, we check whether there are still free RBs, stopping the whole procedure if all frequency resources have been allocated.
At the end, for each user having new feedback data, the current best achievable rate is pushed in his UserHistory.
In Figure 3-9 we report the pseudo code of *Fairness* step.

```
Fairness function
Let Users be the list of all backlogged users
|      for all $h \in Users$ do
|      |      Let $L_h$ be the list of UE h’s List of Pair elemen.
|      |      Let $UH_h$ be the UserHistory of user $h$
|      |      for all ($LSB,h,k,s$) ∈ $L_i$ do
|      |      |      if $b$ is already allocated then
|      |      |      |      continue
|      |      |      end if
|      |      |      if check($UH_h$, $s$)=true then
|      |      |      |      lte_allocation_function($LSB$, $h$, $k$)
|      |      |      end if
|      |      |      break
|      |      end for
|      |      if All LSB have been fully allocated then
|      |      |      break
|      |      end if
|      end for
|      Update UserHistory for each user having new feedback
end function
```

Figure 3-9: Fairness step pseudo code

3.5.3 Efficiency Step

Efficiency step allocates remaining frequency resources following a *MaxCI-like* behavior. The previous two steps should guarantee that we meet system requirements in term of timing constraint and fairness, so this final step aims at maximizing throughput.
If there are not urgent data and no user is experiencing a local peak, the previous steps do nothing, so all frequency resources are allocated considering only system efficiency. This throughput optimal approach is useful to move forward the saturation threshold for the cell: by this we mean that the offered load that the scheduler can manage is higher than that of other scheduling disciplines, like Proportional Fair or Deficit Round Robin.
On the other hand, being *Efficiency* only the last of three steps, the scheduler behavior is not affected by the well known problems of *MaxCI.* A simple opportunistic scheduler, in fact, has no notion of QoS and is unfair with user

experiencing bad channel conditions. Our design, instead, aims at keeping some of the advantages of opportunism, supporting QoS and fairness at the same time.
The procedure implementing *Efficiency* step is the same reported in section 0.

3.6 Performance evaluation

In this section we report the description of the simulator used to evaluate the proposed algorithms, the description of the simulated scenarios and the results of the simulation campaigns.

3.6.1 Simulator

In this paragraph we describe the developed LTE simulator (based on OMNeT++) used to asses the performance of the proposed algorithms.

3.6.1.1 OMNeT++

OMNeT++ [7] is an object-oriented modular discrete-event network simulation framework. OMNeT++ itself is not a simulator of anything concrete, but it rather provides infrastructure and tools for writing simulations. The main features of OMNeT++ framework are explained below.

- It is *modular:* network models are built up by assembling reusable components termed *modules.* Well-written modules can be combined together to form complex models.
- It supports a *discrete-event* simulation paradigm: system behavior is modeled as a chronological sequence of events, which occur at an instant in time and mark a change of state in the system, possibly creating new events.
- It is *object-oriented:* within this framework simulators are developed using C++ [26] and NED (NEtwork Description) programming languages. Both these languages widely support the object- oriented paradigm.
- It is *open-source:* OMNeT++ is free for academic and non-profit use. For commercial purposes one needs to obtain the OMNEST license [26].
  OMNeT++ has a generic architecture, so it can be used in various problem domains:
  1. Modeling of wired and wireless communication networks.
  2. Protocol modeling.
  3. Modeling of multiprocessors and other distributed hardware systems.
  4. Evaluating performance aspects of complex software systems.

In general, OMNeT++ can be used for the modeling and simulation of any system where the discrete-event approach is suitable, and which can be conveniently mapped into entities communicating by exchanging messages.
We've chosen it as the starting point for our LTE and LTE-A network simulator also because of it is currently gaining widespread popularity as a network simulation platform in the scientific community as well as in industrial settings, and building up a large user community.

3.6.1.2 LTE OMNeT++ Embodiment

The LTE simulator ha been build on top of OMNeT++ framework and making use of the INET additional packages. This framework provides:

- the Internet Protocol stack support to OMNeT, needed for simulating the LTE stack application and
- transport layer
- support to simulating radio channels
- network nodes mobility

Using such tools we implemented the following modules:

1. The *Binder*, whose role is to dynamically building and keeping track of simulated network topology, therefore making it available to all other modules requesting topology information for their operations during simulated events.
2. The *World* and the *ConnectionManager* modules, required by INET framework for managing UEs' mobility and handling events related to wireless channel transmission.
3. The *lteInternet*, modeling the Internet and the applications connected to the simulated LTE network.
4. The *eNodeB*, which models the Enhanced Node B of LTE and LTE-A network.
5. The *UE*, which models the LTE and LTE-A User Equipment.
6. The *Relay* module, specific for LTE-A network, and modeling the LTE-A Wireless Relay node, required for the following illustrated work on Relay Link scheduling.

The developed OMNeT++ LTE framework is thus capable of executing system-level simulations of LTE networks, which can be comprised of one or more eNodeB, with full support to real application traffic simulation, on top of either UDP or TCP transport protocol. From the physical layer point of view, it supports all LTE and LTE-A MIMO transmission modes, including MU-MIMO, full OFDMA and SC-FDMA channel simulation with frequency diversity characterization.

3.6.2 Schedulers

In the simulator we implemented several scheduling disciplines, in order to compare the scheduling framework we proposed (hereafter QoSMuMIMO[33][42]*)* with existing schedulers. Beside *QoSMuMIMO,* whose detailed description is given in Chapter 3, we implemented the following scheduling disciplines:

- *MaxCI*
- *MaxCIP* (MaxCI with Priority)
- *PF* (Proportional Fair)
- *PFP* (Proportional Fair with Priority).

MaxCIP and *PFP* schedulers differ from their *priority-unaware* versions because they serve users in descending order of class priority. Users belonging to the same class of service (and thus having the same priority), are sorted (and served) according to the score typical of the scheduling discipline considered: channel quality for *MaxCIP* and Proportional Fair score for *PFP.*

To have a fair comparison with *QoSMuMIMO,* all the schedulers take advantage of both the TMS algorithm (0) and the *proposed* allocation function (0). Therefore these schedulers are actually enhanced versions of the homonymous schedulers available in literature. In fact they can select the transmission mode considering Spatial Multiplexing, Transmit Diversity and MU-MIMO, they are able to allocate users in the frequency domain exploiting frequency selectivity and they ensure that LTE constraints (i.e., one-MIMO-mode, MU-MIMO-pairing, one-MCS) are met. For this reason we append the "++" suffix to the name of these schedulers hereafter (MaxCI++, *Pf++, MaxCIP++, PFP++)*.

3.6.3 Traffic

In our simulations we used four different classes of service, assuming that all the users belonging to the same class have the same traffic and a specific traffic is always mapped on a specific class of service. Therefore there is a *one-to-one* mapping between a traffic type and a class of service, so we will use the words *traffic* and *class* interchangeably. The traffics we considered are the following:

- Real-Time Gaming
- VoIP (Voice over IP)
- VoD (Video on Demand)
- Background Traffic (full-buffer)

To implement Real-Time Gaming we refer to the traffic model proposed in [28].
Since in our study we consider only the downlink direction, we refer to the parameters describing the Server-to-Client traffic. Both packet inter-arrival time and packet size are modeled with *Extreme* distribution. Its cumulative distribution function is given by:

$$F(x) = 1 - e^{-e^{(x-a)/b}}$$

Here a is the *location* and b is the *scale* parameter. Random variables for this distribution can be easily generated by using the inverse transformation as follows:

$$x = a + b\ln(-\ln(u))$$

Here *u* is a random variable uniformly distributed between 0 and 1. The random values generated using the above relation should be rounded to the nearest integer and should be ignored if lower than 0.
The values used for *location* and *scale* parameters are given in Table 3-2.
VoIP traffic [38] is modeled as an ON/OFF process, with silence and talk spurt periods distributed according to Weibull distributions. The packet size and generation interval are selected according to the GSM AMR specifications, which is one of the most employed codecs in wireless cellular networks.
A VoD source consists of a traffic trace from a pre-encoded MPEG4 file [27][39][36], with randomized starting offset.
Finally, background traffic is modeled as an uninterrupted source of fixed-size (4KB) packets to emulate asymptotic conditions, so as to derive the capacity limits of networks and to produce maximum interference to real-time traffic. The configuration values of the traffic models are reported in Table 3-2. In our study we consider only downlink direction.

Parameter			Value
VoIP	Rate (during talkpurts)		12.8 kb/s
	Talk spurt period (Weibull)		l = 1.423 s, k = 0.824
	Silence period (Weibull)		l = 0.899 s, k = 1.089
VoD	MPEG4 trace		Futurama (medium quality)
	Mean (peak) rate		0.28 MB/s
Gaming	Packet inter-arrival time (ms)	Location (*a*)	39.7
		Scale (*b*)	1.9
	Packet size (byte)	Location (*a*)	126.9
		Scale (*b*)	20.4

Table 3-2: Configuration of traffic models

3.6.4 Metrics

For completeness, here we provide the whole list of metrics whose values are available at the end of each simulation, without the need of any further post-processing operation. The common metrics are listed below:

- Cell throughput
- User throughput
- Percentage of selected transmission mode.
- Mean Opinion Score
- User Average Delay

Mean Opinion Score (MOS) is a performance index [31] between 1 (unbearable quality) and 5 (best quality) for VoIP traffic, which combines together the delay and packet loss of VoIP frames.

3.6.5 System Model

The system model parameters are reported in Table 3-3.

Parameter	**Value**
Carrier Frequency	2.0 GHz
Total Bandwidth	20Mhz (10Mhz DL, 10 MHz UL)
Duplexing mode	Frequency Division Duplexing
Channel specifications	ITU URBAN MACRO-CELL
Fast Fading Model	Jakes Fading
Number of tap channel	6
MS distance	100 m, 300 m
MS speed	1 km/h ,120 km/h
eNodeB transmission power	46 dBm
Thermal noise level	-104 dBm
Cable loss	2 dB
Antenna Gain	18 dBi
Noise Figure UE	7 dB
Noise Figure eNodeB	5 dB
Shadowing std. deviation	8 dB
Mobility Model	Circular
CQI type	ideal
CQI reporting interval	4 frames
MCS reporting target BLER	0.1
Frame size	1 ms
No. of Resource Block	50
No. logical bands	5 (10 RBs each)
Max no. H-ARQ Rtx	4
K parameter of TMS	2
RLC PDU size	40 B

Table 3-3: System model parameters

3.6.6 Results

Every scenario was simulated separately for each scheduler. For each combination of parameters we ran 10 independent replications, whose samples were averaged to obtain an estimated mean and 95% confidence interval, which is reported in the figures, unless negligible. The duration of each replica is 200 seconds long. The simulation scenarios differ for the traffic pattern (Gaming, VoIP and VoD) and the channel quality. The combination of speed and distance of the MSs remains the same. Only a selection of the most relevant results obtained are reported. As regards the channel conditions, for each class of service 50% of the users have *bad* channel conditions (400m distance from BS with a speed of 120 Km/h) while the remaining 50% of the users have *good* channel (100m distance from BS with a speed of 1 Km/h). In the following we use the terms bad and good indicating MSs respectively with bad and good channel conditions.
The number of UE for each traffic class is reported in Table 3-4.

Traffic Model	Number of UE
Real time gaming	30 (15 good, 15 bad)
Voice over IP	30 (15 good, 15 bad)
Video on Demand	30 (15 good, 15 bad)
Full Buffer	30 (15 good, 15 bad)

Table 3-4: Simulated scenario

3.6.6.1 QoSMuMIMO Tuning

As described in section 3.5, *QoSMuMIMO* is composed of three different steps: *Urgency, Fairness* and *Efficiency.* Both *Urgency* and *Fairness* steps have tunable parameters for the various classes of service. These parameters are listed below for the sake of completeness.

- *Urgency* step parameters:
  - *Priority (P)*
  - *Delay Budget (D)*
  - *Slack Time (S)*
  - *Max Urgent Burst (M)*
- *Fairness* step parameters:
  - *Window Size (k)*
  - *Threshold (t)*

The values used for *Urgency* step are given in Table 3-5.

Traffic	P	D(ms)	S(ms)	M
Gaming	0	30	6	infinite
VoIP	1	50	10	infinite
VoD	2	500	100	infinite

Table 3-5: Urgency settings for each traffic class

Priority and Delay Budget values are chosen using as reference the LTE QCI (QoS Class Identifier) table given in [29]. As regards Slack values, we use a time of 20% of the Delay Budget which allows *urgent* data to be sent without missing the deadline. Max Urgent Burst values is ideally set to infinite to avoid limits on the maximum amount of data served by the *Urgency* step. We highlight that the *Full buffer* class does not appear in Table 3-5, since the corresponding traffic has no *real-time* constraints, so it is never served by *Urgency* step.
In order to evaluate the impact of the *Fairness* step we define three different setup for this step that we called *Fair1, Fair2, and Fair3.*
The values we used for these setup are given in Table 3-6.

Traffic	Fair1		Fair2		Fair3	
	t	k	t	k	t	k
Gaming	30	20	30	25	30	29
VoIP	30	20	30	25	30	29
VoD	30	20	30	25	30	29
Background	30	20	30	25	30	29

Table 3-6: Fairness step settings

The window used to keep track of the channel quality of each user over the time has the same size for all the classes of service. It is straightforward that with *Fair1* setup fairness should be higher than *Fair3*.
In order to help the interpretation of the simulation results we use the same template for all graphs. In particular we show for each metric the results for single UE (both *good* and *bad)* and the average of all the users regardless their channel quality. The results are grouped according to the scheduler and simulation settings. in this section we focus on the comparison between the results obtained simulating the proposed scheduler and the three fairness configurations, Fair1, Fair2 and Fair3.
In Figure 3-10 we report the Mean Opinion Score. We observe that the settings of fairness step do not affect the performance provided by the scheduler, i.e. all the configurations provide comparable results.

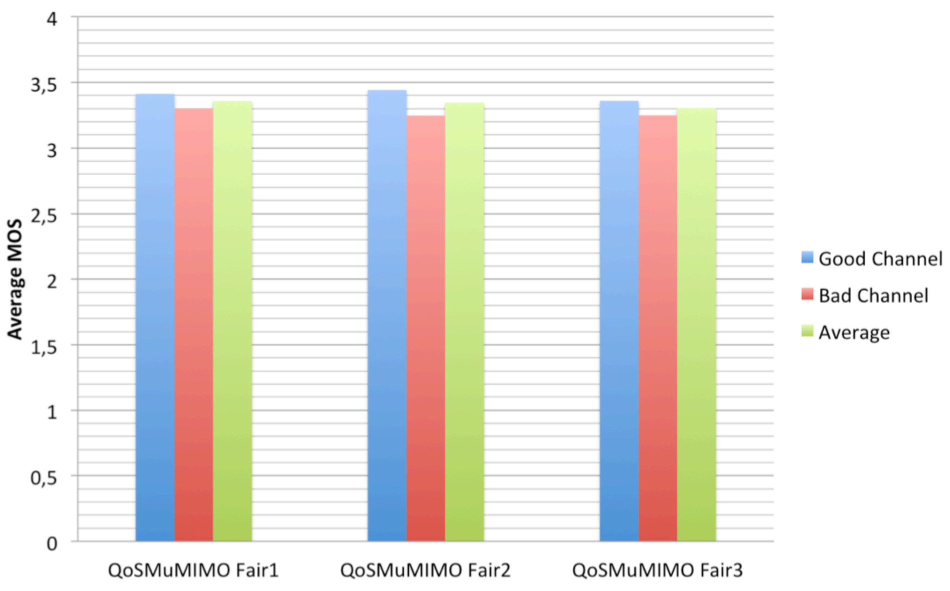


Figure 3-10: Average MOS of UE with different fairness settings

Figure 3-11 reports the average delay of UE with gaming traffic. As can be seen the same behavior is confirmed.

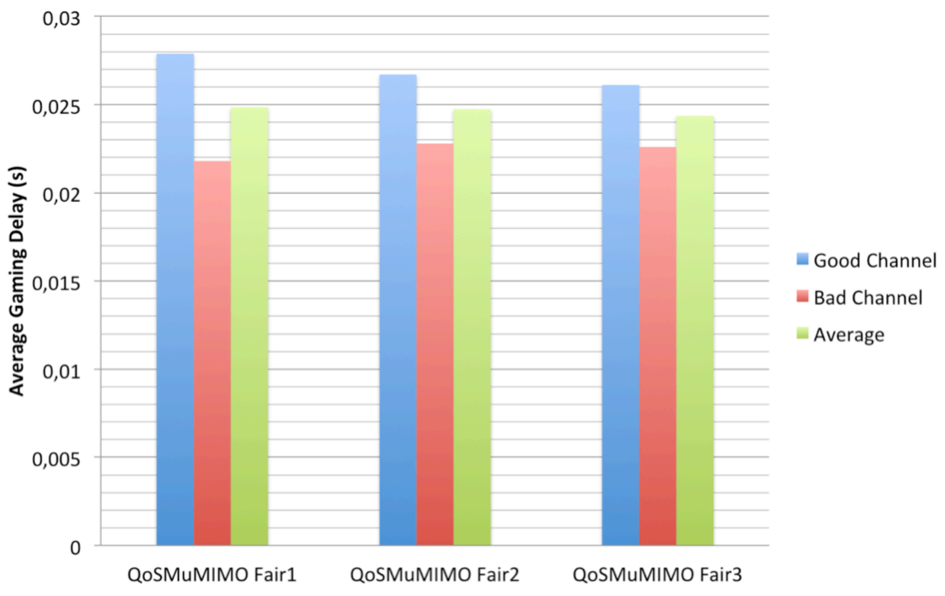


Figure 3-11: Average delay of gaming traffic with different fairness settings

In Figure 3-12 is shown the average delay per UE with video on demand traffic. In this graph we can observe that *Fair1* configuration offers an high level of fairness, i.e. good users and bad users experience a comparable delay. On the other hand with *Fair2* and *Fair3* configurations bad users have a delay which is 70% higher than good users. This can be explained with the lower fairness offered by these configurations. We can also notice that the average delay of the three configurations is similar.

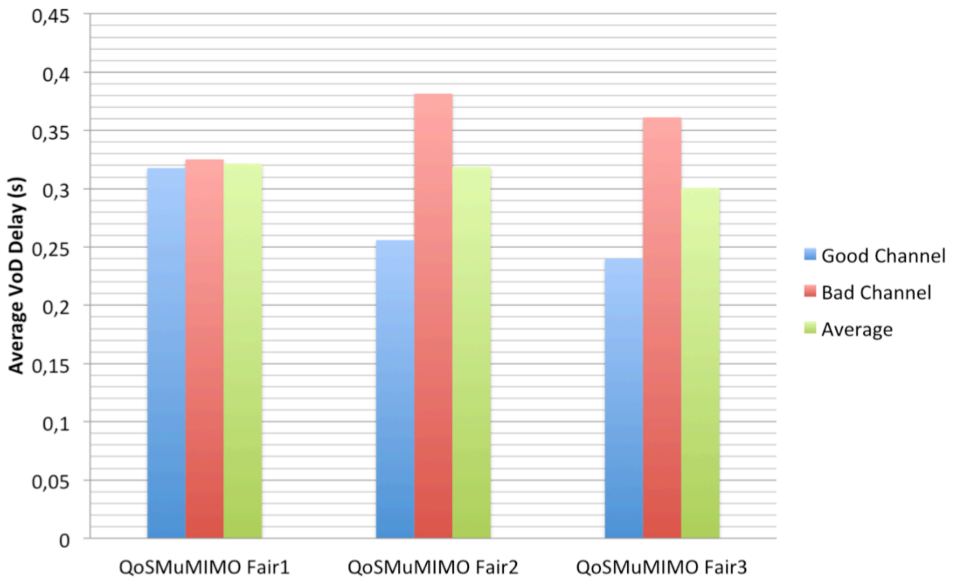


Figure 3-12: Average delay of Video on Demand traffic with different fairness settings

Finally in Figure 3-13 is reported the average throughput of the full buffer users. In this metric we observe the real impact of the *fairness step* in the system performance. In particular with *Fair1* configuration users with good channel and users with bad channel experience comparable performance. On the contrary, with *Fair2* configuration, users with good channel have an higher throughput than the users with bad channel. This behavior is even more pronounced with the *Fair3* configuration. We can also note that going from Fair1 to Fair3 bad users decrease their throughput while good users increase their performance. This is because *Fair3* is a configuration that provides an higher spectral efficiency paid in terms of fairness. *Fair1*, on the other hand, offers higher fairness which is paid in terms of efficiency. This is supported comparing the average value in Figure 3-13 that show an increasing throughput from *Fair1* to *Fair3*. *Fair2* offers a tradeoff between efficiency and fairness.

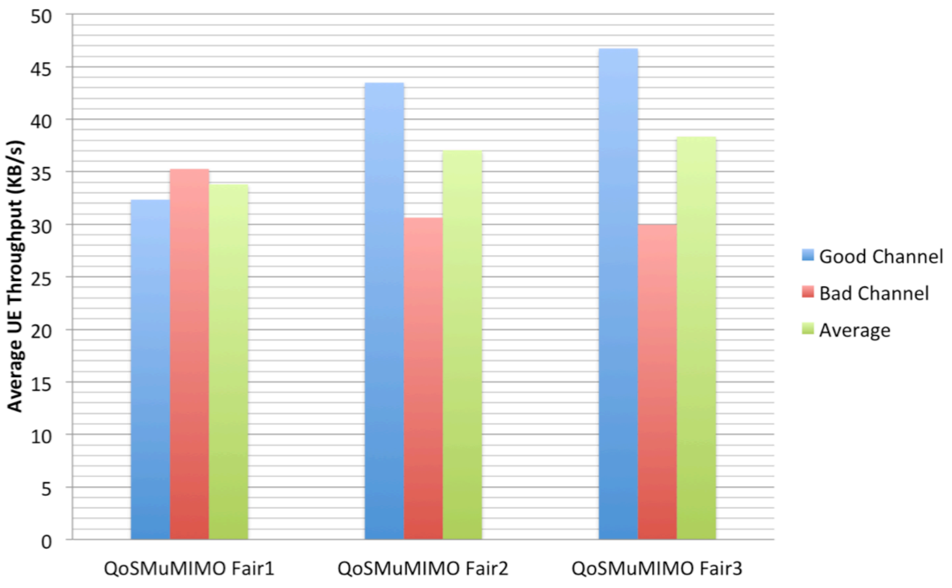


Figure 3-13: Average Throughput of full buffer UE with different fairness settings

Figure 3-14 shows the percentage of usage of each transmission mode. We do not report the results for each fairness setup because as described in section 0 the TMS algorithm works on top of the scheduler and consequently its results do no depend on the scheduler or its settings. In particular we can note that spatial multiplexing is used most of the time. Spatial multiplexing and *MU-MIMO* offer higher spectral efficiency than transmit diversity thanks to the simultaneous transmission of the codeword. This is the reason why transmit diversity is used only in 13% of the cases. By the way *SpMux* is used more often than *MU-MIMO* because when an UE reports an high CQI the probability that also an high CQI for the second codeword is high. On the contrary with spatial multiplexing it is more difficult find two UE with data to transmit that can be paired and both with high CQI.

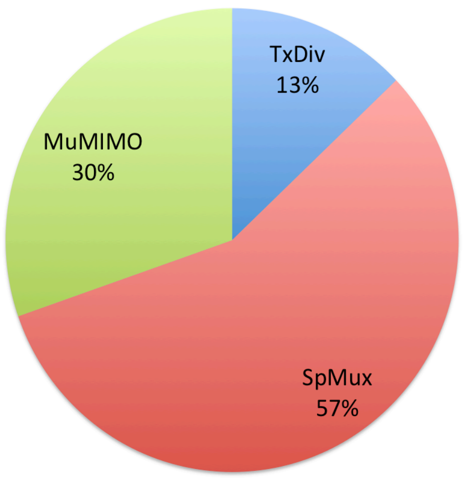


Figure 3-14: Transmission mode usage

3.6.6.2 Comparison with other scheduler

Here *QoSMuMIMO* is compared with the schedulers described in 3.6.2. First of all we consider the mean cell throughput, shown in Figure 3-15

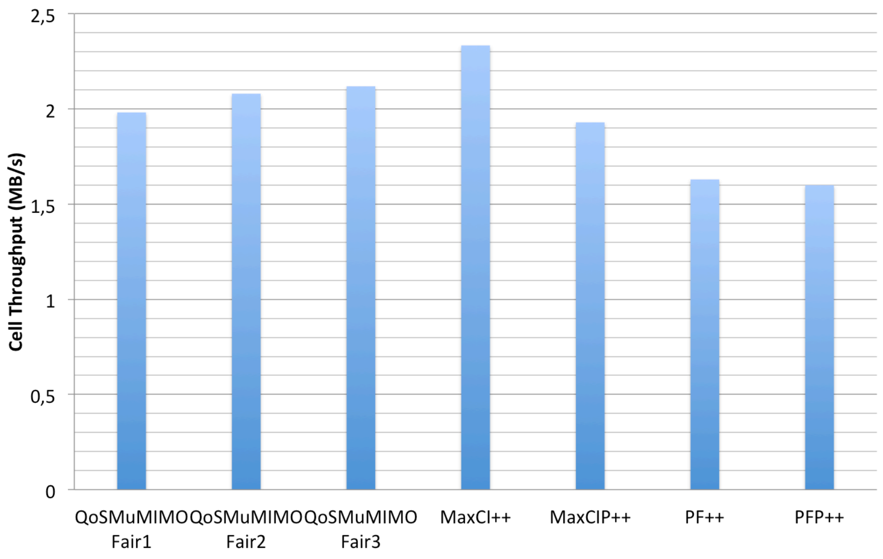


Figure 3-15: Cell throughput with different scheduler

As we could expect, *MaxCI*++ achieves the best performance because its objective is to maximize the cell throughput. *QosMuMimo* follows, experiencing a throughput loss respectively of 13% (Fair1 configuration), 7% (Fair2 configuration) and 4% (Fair3 configuration) with respect to *MaxCI*++. All the other schedulers exhibit a lower performance, with a throughput loss between 20% and 35%. *Pf*++ sacrifices efficiency in order to achieve system fairness. Priority schedulers (MaxCIP++ and

PFP++) suffer from the high number of users belonging to the highest classes of service: if there are backlogged users with high priority, these schedulers are forced to serve them, even if their current channel conditions are bad.

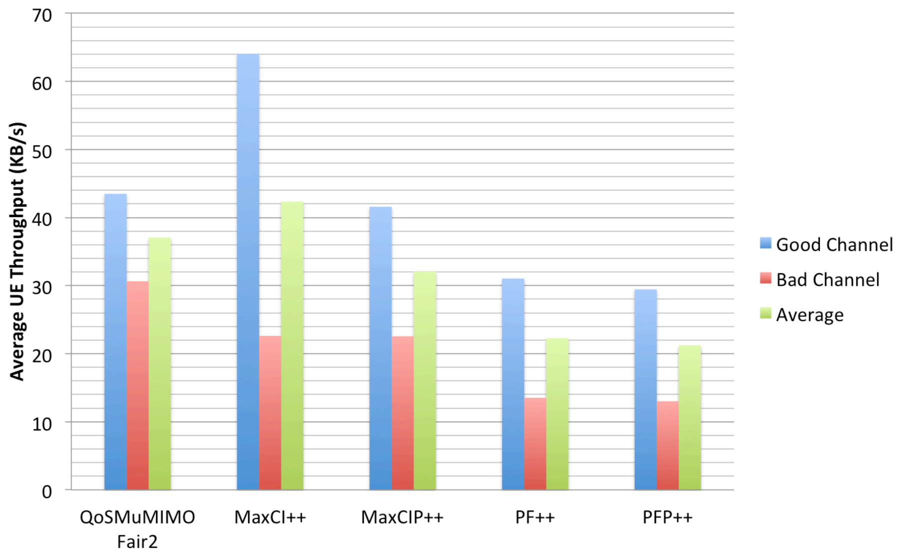


Figure 3-16: Average Full buffer traffic throughput with different scheduler

We analyzed in detail the performance of *QoSMuMIMO* with different fairness settings in section 3.6.6.1. For the sake of simplicity, we decide, from now, to show only one fairness configuration. We decide for Fair2 because offers the best tradeoff between efficiency and fairness.
In Figure 3-16 shows the average throughput of an UE with full buffer traffic. We note that *MaxCI*++ offers best performances, particularly for UE with good channel. Moreover *MaxCI*++ provides also an unexpected results: high throughput to users with bad channel. The legacy *MaxCI* scheduler (without TMS) aims at minimizing the efficiency causing also starvation for users with bad channel. With *MaxCI*++ this behavior is not shown thanks to the TMS and mostly thanks to the *MuMimo* transmission mode. In particular *MuMimo* transmission mode increases the fairness of the systems because with the same time-frequency resource two UEs are served, consequently the number of UE served by the system in the same TTI is higher than using other transmission modes.
In terms of average throughput *QoSMuMIMO* performs well, second only to *MaxCI++,* all the other schedulers provide lower performance. This result confirms what we have already seen in Figure 3-15. Furthermore QoSMuMIMO offers best performance for bad UEs. This means that with *QoSMuMIMO*, users with bad channel, typically at cell edge, are served with higher efficiency providing also higher efficiency. This is ensured by the TMS algorithm and fairness step.
Comparing priority version (MaxCIP++ and PFP++) with normal version (MaxCI++ and PF++) we observe that the priority version perform worst in terms of throughput. This is because the priority version schedulers are forced to serve always high priority users even if they are experiencing bad channel conditions,

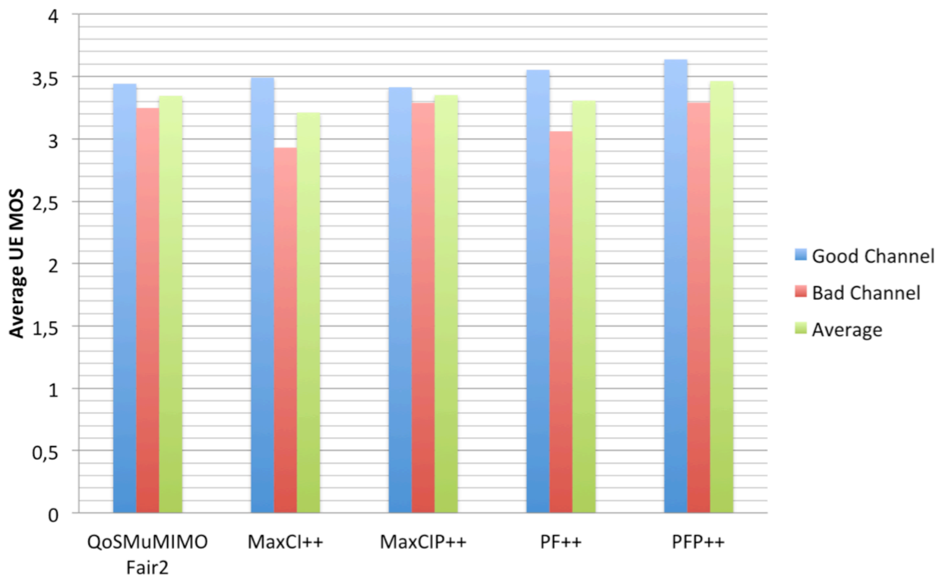


Figure 3-17: Average MOS of Voice over IP traffic with different scheduler

Being in a system with different traffic types, throughput is only an aspect of the whole system performance. In fact, when traffic with *real-time* constraints are considered, it is of paramount importance that deadlines are met . For this reasons we report in Figure 3-17 the average Mean Opinion Score experienced by each UE. As we can see all the schedulers perform well for different reasons:

- *QoSMuMIMO* offers good performance assured by its *urgency* step
- *MaxCIP*++ and *PFP*++ serve real time traffic with strict priority rather than other traffic
- *MaxCI*++ for the same reasons expressed above: the usage of MuMimo transmission mode increases the possibility of being served. Considering the low bit rate generated by a VoIP encoder, only a few RBs are sufficient to ensure good performance.
- *PF*++ tends to serve more often traffics with low bit rate.

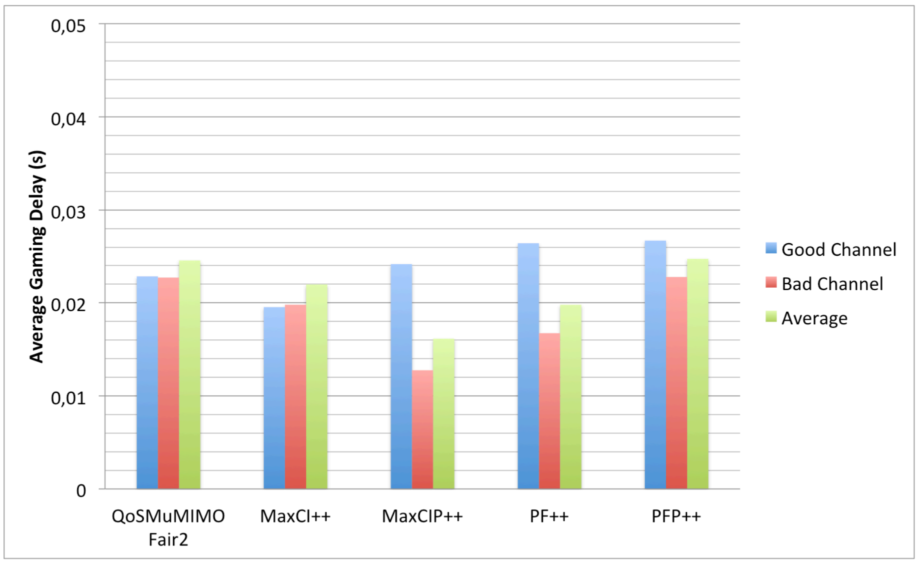


Figure 3-18: Average delay of gaming traffic with different scheduler

The exact same behavior can be seen in Figure 3-18 where we report the average delay of users with gaming traffic: all the schedulers performs well for the same reasons explained above. In particular with this traffic *PF++* performs slightly better than the other schedulers.

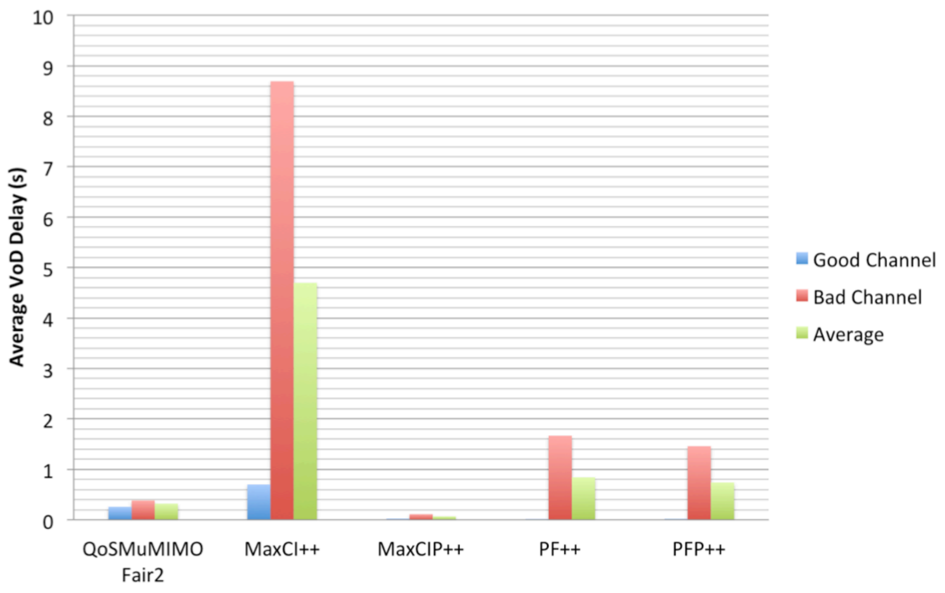


Figure 3-19: Average delay of Video on Demand traffic with different scheduler

Finally, in Figure 3-19 is shown the average delay of Video on Demand traffic. For the sake of simplicity, we report in Figure 3-20 a rescaled version of the same graph. With this traffic class *MaxCI++* and *PF++* offer very bad performance (delay

significantly higher than 1 second). As a matter of facts, this traffic class has a higher bit rate than gaming and *VoIP*, consequently for *VoD* traffic, unlike what we saw in Figure 3-17 and Figure 3-18, the usage of MuMimo as transmission mode is not sufficient to ensure good performance.
Only QosMuMimo and MaxCIP++ result in good performance. Regarding QosMuMimo we can note that *urgency step* accomplish its purpose, in fact the average delay of video on demand does not exceed the *Delay Budget* set for this traffic (Table 3-5).On the other hand MaxCIP++ performs well because it serves video on demand with strict priority than other traffic.
Being forced to serve always high priority regardless their channel conditions, *PFP*++ is less efficient than *the other scheduler* in the use of frequency resources (i.e. it tends to use lower MCS than *PF++),* as confirmed by throughput results in Figure 3-16. The massive use of low MCSs makes *PFP*++ slow in removing high priority users from the buffers, thus increasing the waiting time of *VoD* users.

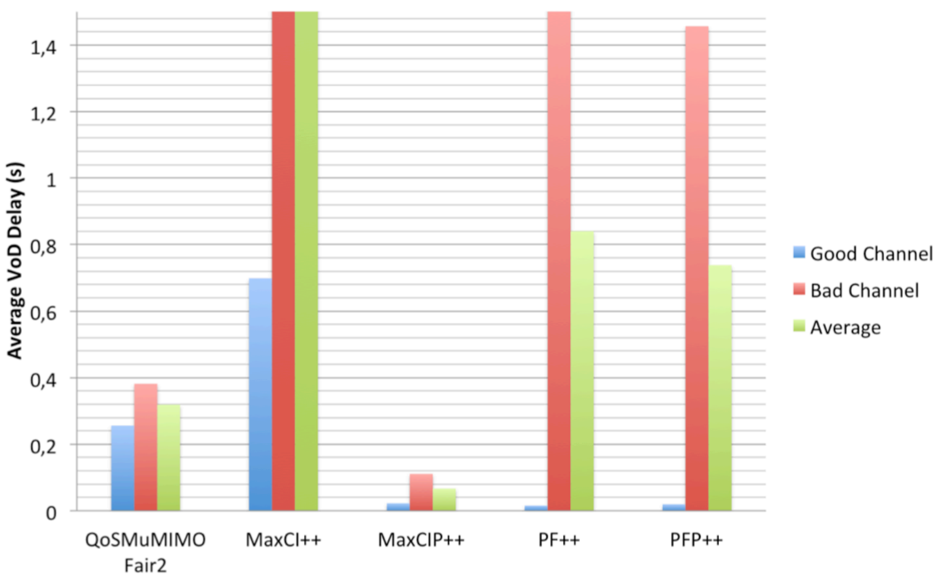


Figure 3-20: Average delay of Video on Demand traffic with different scheduler (rescaled)

3.7 Related Work

As already explained, no work that we are aware of in the literature deals with TMS, resource allocation, fairness and QoS simultaneously. Moreover, the works which consider some of the above aspects often rely on simplifications and idealizations that make them unsuitable for the LTE-A environment. For example, the works considering TMS either ignore MU-MIMO (e.g. 16) or do not obey the *MU-MIMO pairing* constraint (e.g. 17,18). The TMS issue is altogether neglected, assuming *a priori* a single transmission mode in 19-20). As for MCS selection, no solution that we are aware of takes into account the *one-MCS* constraint: it is instead assumed that different MCSs can be used on different RBs, which makes the allocation unusable in practice. We review the above works in more detail.

In 16 authors address the resource allocation problem for UEs using only SU-MIMO modes (i.e. either S-MUX or TD). An optimization problem is formulated, whose objective is to maximize the Proportional Fair (PF) criterion, extended to frequency and spatial domains, under the *one-MIMO-mode* constraint. The above problem is proved to be NP-hard, and two polynomial-time approximations are proposed. While interesting from a theoretical point of view, this work assumes *full-buffer* traffic and different MCSs for the same codeword. Both algorithms are thus impractical. 18 extends the above model including MU-MIMO. The proposed algorithm selects, for each RB, the best user (or pair thereof, for MU-MIMO). As this may violate the *one-MIMO-mode* constraint a *conflict-resolution* algorithm is used a posteriori. However, the latter does not actually prevent a UEs from using *both* S-MUX and MU-MIMO within the same TTI, nor it avoids different UE pairings on different RBs, thus violating both the *one-MIMO-mode* and the *MU-MIMO* constraints.

In 17 authors evaluate several methods to select MU-MIMO pairs (S-MUX being considered as a sub-case) and allocate frequency resources. The considered system has *N* users and *B* RBs. For each RB *b*, authors consider a *N´N sum-rate* matrix, similar to our RM, which is instead *N´*(*N*+1), as it includes TD. However, they neglect both the *one-MIMO-mode* and the *MU-MIMO-pairing* constraints.

In 22 authors propose a joint feedback and scheduling scheme for differentiated-service MU-MIMO systems using Zero-Forcing Beamforming (ZFBF) as a precoding technique. The whole channel matrix *H* should be fed back by each UE to achieve ZFBF, hence authors aim at reducing such reporting burden, while supporting two different classes of service. The eNodeB is supposed to have *k* transmit antennas and at each transmission time at most *k* users can be served, spatially multiplexed, since FDMA is not considered. In LTE systems, precoding is performed using a codebook based unitary precoding 23, also known as Per User Unitary and Rate Control (PU^{2}RC), which differs from ZFBF, so the solution proposed in this work is not applicable.

Another proposal for QoS support in MU-MIMO systems is in 20. In this paper the PF scheduler is extended to support joint space-time-frequency scheduling and QoS. Authors consider a generalized MU-MIMO system, where up to *L* users can be multiplexed in the same RB. QoS support is provided by modifying the well-known PF score according to two parameters which are a characteristic of the service: a data rate weight c_k and a time window w_k, representative of the tolerable delay. For each possible group of *L* users and for each RB the aggregated score is computed by summing the *L* individual scores, and the best

group of *L* users on each RB is selected. Only MU-MIMO is considered (S-MUX and TD are not), and the proposed algorithm neglects the *MU-MIMO-pairing* constraint.
In 24 the well-known PF scheduler is extended in order to support QoS. The reference scenario is Virtual MIMO (VMIMO) system, i.e. the one used in the uplink of LTE. Simulation results show that MU-MIMO improves throughput and fairness also in the uplink.
In 19 different scheduling policies for improving call capacity for VoIP traffic in MIMO-OFDMA networks are considered. TMS is however left out of the picture, as S-MUX is always assumed. The scheduling disciplines analyzed are *Maximum Rate, Proportional Fair, Relative Strength* and *Urgency Based* scheduling, which are tested on VoIP only. Via numerical results, authors show that a fairness-based approach fails to guarantee service for VoIP, while the best QoS is achieved via a greedy scheduling approach that serves packets faster. Due to VoIP packets being small, a packet transmission in good channel conditions requires few resources. Therefore, the overall fairness is maintained by prioritizing strong users requiring fewer resources, hence allowing weaker users to access more resources, which increases the likelihood that they will meet their deadline.
In 21, a *Score-Based* (SB) scheduler is proposed, which achieves fairness with a slightly different mechanism from ours. Instead of selecting a user when its transmission rate is high compared to its own average throughput (as done by the PF scheduler), SB selects a UE when its *score* is high: the score at time t is computed as

$$\sum_{t-W \le x < t} 1_{\{CQI(x) < CQI(t)\}}$$

i.e., the number of past instants, over a window of *W* TTIs, when the CQI was below the current one. UEs are served by decreasing score. Our approach is different, as UEs are only served in the fairness step when their channel quality is particularly good, and with as few resources as strictly required to achieve fairness, whereas the bulk of the resources is assigned in the opportunistic step.

4 ENERGY AWARE PROVISIONER

4.1 Introduction

The present chapter introduces an algorithm for an LTE-Advanced cell in order to increase the energy efficiency.

This algorithm can be seen as a resources provisioner that works on top of the scheduler presented in chapter 3. The objective of this algorithm is to:

- compute the amount of resources (i.e., resource blocks, RBs) required in a superframe (i.e., a set of 40 consecutive TTIs) required to clear the cell backlog in the downlink direction
- decide which subframes should be activated in order to use those RBs at the minimum power, and how many RBs should be made available to a scheduler in each subframe.

Furthermore, we explore possible out-of-standard solutions, where it is assumed that the eNodeB can be activated/deactivated TTI by TTI, i.e. without employing the concept of superframe.

4.2 MBSFN patterns in LTE-Advanced

MBSFN operation is a feature originally devised for multicast transmission in LTE, now reused for intra-eNB energy saving purposes. In fact, in the first place MBSFN subframes have fewer common reference signals (CRS) than normal subframes, hence configuring as many as possible MBSFN subframes allows reduced eNB transmission time. According to current specifications, at most 5 (resp., 6) MBSFN subframes can be configured per radio frame for TDD (resp., FDD). Moreover, during MBSFN subframes sleep modes within the eNB can also be triggered, thus enabling additional and more significant energy savings. In the context of the present work the impact of a potential dynamic MBSFN switching mechanism should be investigated in terms of energy savings achievable by means of the introduction of sleep modes within the eNB.

Note that not all of the subframes can be skipped in order to remain backward-compliant to Rel-8 UEs attached to the eNB. In particular:

- Subframe #0 and #5 are used for SCH and BCH transmission in Rel-8;
- Subframe #4 and #9 are used for paging transmission in Rel-8.

The figure below shows which subframes are eligible for MBSFN operation. The uneligible subframes will be termed *pinned* subframes from now on, and the eligible ones (those where a decision has to be made on whether activating them or not) will be termed *free*. A *free* subframe can be either *active* or *inactive*, whereas a pinned one is active by default.

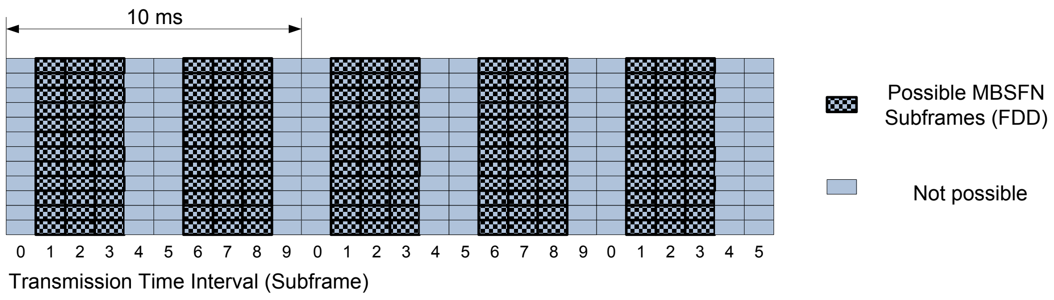


Figure 4-1: LTE-A – Allowed MBSFN subframes

Regarding the MBSFN operation in Rel.8 eNB and related time granularity, MBSFN subframes can be signaled via RRC within a period of 4 radio frames (corresponding to an interval of 40 ms). In the current work (focused on LTE but also in LTE-Advanced features) the impact of a different mechanism (transparent to the standard) enabling dynamic MBSFN subframes switching (e.g. for Rel. 10 terminals, while keeping backward compatibility with legacy terminals) will be considered from the point of view of the packet scheduling operation.

4.3 System model and problem formulation

QoSMuMIMO (Chapter 3) allocates resource for a set of UEs by allocating up to N Resource Blocks (RBs) to the latter. The scheduler can be activated on each TTI. The activation of the eNodeB transmission is modulated by *MBSFN superframes*. The latter are strings of T bits, $a_0,...,a_{T-1}$ (it is usually $T=40$). Without loss of generality, we count time so that MBSFN superframes are aligned at multiples of T. When a MBSFN superframe is superimposed on a cell with a downlink scheduler, the transmission logic is activated at some time instants. times $a_i \cdot m \cdot T$, where $a_i = 1$, and put to sleep at times $a_j \cdot m \cdot T$, where $a_j = 0$, for any $m \geq 1$. MBSFN superframes can be changed instantaneously at multiples of T. The eNodeB is considered to be always receiving, no matter whether its transmission logic is switched on or off. For ease of reading, we will write that the eNodeB is either *active* or *inactive* at time t if $a_{t \bmod T} = 1$ or $a_{t \bmod T} = 0$ respectively. a_i variables are equal to 1 for pinned subframes.
An eNodeB exhibits a different power consumption based on whether it is active or inactive, and – if active – based on how many RBs it is employing for transmission. The power consumed in *inactive* subframes is constant and equal to $P = P_{off}$. The power consumed in *active* subframes, although depending on the type of subframe, is an *affine* function of number of allocated RBs, i.e, $P = P_{base} + \rho \cdot n$, where $P_{base} \geq P_{off}$ and $n \leq N$ is the number of allocated RBs.

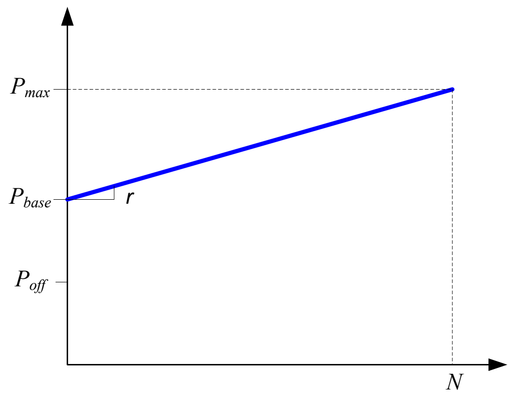


Figure 4-2: Power model

As anticipated, the power consumed in an *active* subframe depends on the type of subframes. In pinned subframes, some power is consumed (even if no RBs are allocated) to transmit control channels (i.e., broadcast, paging, synchronization), whereas those channels need not be transmitted in free subframes. As a consequence, the *baseline power* P_{base} is going to be larger for pinned subframes than for free subframes. It is also obvious that the *maximum* power that can be consumed in an active frame (when N RBs are allocated) does *not* depend on the presence of control channels, and it is equal to $P_{\max}$. Hence, the *per-RB power consumption rate* ρ depends on the baseline power as well through the following formula:

$$\rho = \frac{P_{\max} - P_{base}}{N} \qquad (1)$$

Finally, the *exploitable per-RB capacity* depends on the type of frame. In fact, in pinned subframes, the presence of control channels eats out some bits that could otherwise be available for transmitting UE traffic. We denote with x the exploitable per-RB capacity of an active subframe, $0 \le x \le 1$. For instance, it is $x = 1$ for free subframes.

Henceforth, we will define as *equivalent* two subframes whose power model and exploitable per-RB capacity are equal. There are, in fact, four equivalence classes of subframes: broadcast subframes, synchronization subframes, paging subframes, and free subframes. The first three are pinned. In order to simplify the notation, we define the following sets of indices $\Pi = \{b, s, p\}$ and $\Phi = \{f\}$, and use $i \in \Pi \cup \Phi$ as a generic index. Hence, we denote with T_i the number of subframes in equivalence class i, with $\sum_{i \in \Pi \cup \Phi} T_i = T$, with $P_{base,i}$ the baseline power, with ρ_i the related per-RB power consumption, and with x_i the exploitable per-RB capacity.

The problem that we address in this document is to minimize the overall power consumption over a superframe, i.e. $\sum_{t=0}^{T-1} P(t)$, given the (unknown) arrivals at the eNodeB. Obviously, this problem can be addressed at two levels (not mutually disjoint): one is the *scheduling* level, where a more efficient scheduler will transmit the same quantity of traffic using fewer RBs than its competitors. Another one is the *activation* level, which is the focus of the present document. At the activation level, minimizing the power consumption means to decide which subframes are active, and how many RBs the scheduler can be made to allocate in each active subframe.

We first show that, if the traffic demand for a superframe is *known in advance* and equal to K RBs, then the problem can be formulated as an integer-linear optimization problem. While solving the latter is indeed possible, a heuristic algorithm can also be designed, which achieves good performance. We then exploit the solution thus computed to extrapolate algorithms that cope with unknown demands.

Strictly speaking, a scheduler transmits *PDUs*, which may (and often do) occupy more than one RB. Therefore, if B RBs are made available to the scheduler in a given subframe, some of them may actually be unusable due to PDU quantization even though a demand exists for them. The maximum number of RBs that a PDU may occupy depends in turn on the channel quality of the UE the PDU is addressed to. When the channel conditions are poor, even relatively short PDUs may occupy a large number of RBs, so that quantization significantly affects schedulability. We initially assume that *each and every* RB is exploitable (i.e., that PDUs are arbitrary divisible), and then show how to cope with fixed-length PDUs later on.

4.4 Approximating the optimal solution for known traffic demands

The problem is to find the minimum-power solution that enables the eNodeB to exploit at least K RBs in a superframe. Being able to *exploit* K RBs is in fact a throughput constraint. The following observations are in order:

it is irrelevant *how* pinned/free subframes are interleaved in a superframe; in fact, switching two different time instants in an MBSFN pattern does not change either the throughput or the power consumption in a superframe.

Likewise, it is irrelevant how *active* subframes are interleaved in a superframe, as long as they cover the required demand. Obviously enough, how active frames are arranged has strong implications on the traffic *delay*. However, we can always permute an active and an inactive subframe without modifying the overall throughput and power consumption. Therefore, given a power-optimal solution that verifies the throughput constraint, we can compute a *delay-optimal* solution with the same power consumption simply by scheduling active frames.

For the above reasons, it is evident that, from both a throughput and a power consumption perspective, only the following variables are relevant:

- for *pinned* subframes, how many RBs are allocated *to each equivalence class of subframes*;
- for *free* subframes, how many RBs are allocated to free subframes, and how many of them need to be active in order to support that number.

As far as i) is concerned, the following property is in fact obvious:

Property 1: given that an overall B RBs in a superframe are allocated to class-i *pinned* subframes, it is irrelevant how they are split among the T_i subframes of their class.
Proof: the per-RB power consumption is constant within a class, hence moving one RB from one subframe to another does not change the cost. The exploitable per-RB capacity is also constant (recall that we have assumed that PDU quantization is not a problem), hence moving one RB from one subframe to another does not change the throughput.

□

As far as ii) is concerned, the following property holds:
Property 2: given that an overall B RBs in a superframe are allocated to free subframes, the minimum-power solution compliant with that allocation is one where $\lceil B/N \rceil$ free subframes are allocated, and it is irrelevant how the RBs are shared among the free subframes (as long as the above figure is not exceeded).
Proof: the per-RB cost of free subframes is constant, and their exploitable per-RB capacity is also constant. Hence moving one RB from an *active* free subframe to another *active* one does not change either the cost or the throughput. On the other hand, since $P_{base,f} \geq P_{off}$, an active free subframe consumes more power than an inactive one. For this reason, the number of active free subframes has to be kept as small as possible. The minimum number of active free subframes required to accommodate B RBs is $\lceil B/N \rceil$.

□

This said, we can model the problem of minimum-power allocation given the throughput constraint as an integer-linear optimization problem as follows. Call b_i the number of RBs allocated to class-i subframes, and let a be the number of active free subframes. The overall power consumption in a superframe is:

$$\begin{aligned} P &= \sum_{i\in\Pi} P_{base,i} \cdot T_i + P_{off} \cdot \left(T - a - \sum_{i\in\Pi} T_i\right) + P_{base,f} \cdot a + \sum_{i\in\Pi\cup\Phi} \rho_i \cdot b_i \\ &= \left[\sum_{i\in\Pi} P_{base,i} \cdot T_i + P_{off} \cdot \left(T - \sum_{i\in\Pi} T_i\right)\right] + \left(P_{base,f} - P_{off}\right) \cdot a + \sum_{i\in\Pi\cup\Phi} \rho_i \cdot b_i \end{aligned} \tag{2}$$

The addenda between square brackets in (2) are constant, hence uninteresting from a decision standpoint. The constraints under which the allocation has to be performed are the following:

$$\begin{array}{ll} b_i \leq T_i \cdot N & i \in \Pi \\ b_i \leq a \cdot N & i \in \Phi \\ a \leq T_f & \\ \sum_{i\in\Pi\cup\Phi} x_i \cdot b_i \geq K & \end{array}$$

Hence, we can compute the minimum-power solution by solving the following integer-linear optimization problem:

$$\begin{aligned}
&\min\left\{\left(P_{base,f}-P_{off}\right)\cdot a+\sum\nolimits_{i\in\Pi\cup\Phi}\rho_i\cdot b_i\right\}\\
&s.t.\\
&\quad b_i\le T_i\cdot N \qquad i\in\Pi\\
&\quad b_i\le a\cdot N \qquad i\in\Phi\\
&\quad a\le T_f\\
&\quad \sum\nolimits_{i\in\Pi\cup\Phi}x_i\cdot b_i\ge K\\
&\quad b_i,a\in Z^+ \qquad i\in\Pi\cup\Phi
\end{aligned} \tag{3}$$

Problem (3) has $|\Pi|+2\cdot|\Phi|$ integer variables, and can be solved by an integer-linear solver. However, a good heuristic solution can also be found algorithmically, which gives more insight into the behavior of the system.

4.4.1 Throughput-oriented heuristic

The key parameter is the power-capacity ratio (PCR) ρ_i/x_i, $i\in\Pi\cup\Phi$. The latter measures how much power it is consumed per exploitable RB in class-i subframes. The right way to proceed would therefore be to allocate RBs only to the class with the smallest PCR, and move to the next smallest only after the former has been depleted. We therefore split the classes of pinned subframes in two sets, L and H. The first set includes those classes whose PCR is smaller than (or equal to) ρ_f/x_f, whereas H includes the classes whose PCR is strictly higher than ρ_f/x_f. The first thing to do is to fill up the subframes in L, going by increasing PCR. Only afterwards free subframes may be considered for allocation. For these, in fact, activating a single subframe has also a *fixed* cost equal to $P_{base,f}-P_{off}$, independent of the number of allocated RBs. Hence, that fixed cost has to be weighted in as well when comparing against allocating the same number of RBs to subframes in H.

With reference to the pseudo code in Figure 4-3, the classes of pinned subframes in L get served first until depletion (lines 2-7). After that, the smallest-PCR class becomes that of *free* subframes, and we must check two conditions (line 10):

- if the traffic that fits in *one* free subframe is larger than the cumulated capacity of *all* subframes belonging to classes in H (unlikely as it may be), or
- if the cost of filling up one free subframe entirely is smaller than the cost of obtaining the equivalent amount of capacity using pinned subframes in H, (lines 8-9),

then the lowest cost solution is to allocate one free subframe. Furthermore, it is clear that this condition is going to hold as long as there is enough capacity to fill one free subframe entirely. Therefore, we can directly activate $d=\min\left\{\left\lfloor K/\left(N\cdot x_f\right)\right\rfloor,T_f\right\}$ free frames at once, allocating N RBs to each, as this is the minimum-cost choice to cover that capacity (lines 11-12). Note that the cycle in lines 8-9 computes the cost of a *virtual* allocation of N entire RBs's worth of capacity at the smallest cost using only pinned subframes in H. In fact, it first

considers the smallest-cost class, and virtually allocates either $N \cdot x_f / x_i$ RBs if $N \cdot x_f / x_i \le T_i$. Otherwise, it virtually allocates T_i RBs, and moves on to the next smallest class in H to allocate the residual amount of RBs, and so on.
After all the whole free subframes have been allocated, either $a = T_f$, in which case free subframes are depleted and we can only clear any residual demand by resorting to pinned subframes in H, or $a < T_f$, in which case there still is a free subframe to give away, which will however not be a full one (being the remainder of ratio $K / (N \cdot x_f)$). In this last case, it remains to be seen if the cost of allocating a *partial* free subframe is justified, which requires a similar check as the one for whole free subframes, mutatis mutandis (lines 15-21).
After free subframes have been considered (and possibly allocated), there may still be some unsatisfied capacity. In order to clear it, we must first allocate RBs to the classes in H (lines 22-25). If at the end there still is some residual capacity, we need to allocate available free subframes, no matter what the cost (lines 26-33).
Now, it may happen that at the end of the algorithm the number of full blocks K is exceeded by more than x_j, for some class j such that $b_j > 0$. In this case (lines 35-37), the power consumption can be reduced by removing the highest-cost RB belonging to the set of the above classes. Note that, since $x_f > x_j \ \forall j \in \Pi$, then the capacity allocated so far is $C \in [K; K + x_f[$. Hence, removing one *free* RB would make the allocation infeasible. Thus we may only constrain ourselves to removing RBs belonging to pinned subframe classes. Furthermore, if $x_j \ge 0.5 \ \forall j \in \Pi$ (which is likely in practice) the while loop at lines 35-37 is executed just once, as at most one RB can be removed.
The complexity of the above algorithm is $O(|\Pi|)$, hence its cost is independent of T, i.e. the number of subframes in a superframe (whereas finding the optimal solution is exponential in the number of variables, i.e. of frame classes).
We show how the algorithm works through a simple example:

Example
Consider the following parameters (numbers are selected for illustrative purpose, and need not be representative of a real-life case):

Subframe	**b**	**s**	**p**	**f**
ρ	12	7	11	10
x	.92	.85	.90	1
ρ_i / x_i	13.04	8.23	12.22	10
T_i	4	4	8	24

With $N = 10$, $K = 100$, $P_{base,f} - P_{off} = 20$.

We have $L=\{s\}$, and $H=\{b,p\}$. Hence, the first step is to allocate all the RBs in class-*s* frames, i.e. $4\cdot10=40$. This covers a capacity equal to $0.85\cdot40=34$. After that, we move to considering *free* subframes. The cost of activating a free subframe is $P_{base,f}-P_{off}+N\cdot\rho_f=120$. On the other hand, an equivalent capacity of 10 RBs can be obtained using RBs in H if we allocate *twelve* RBs from class-*p* frames (a total of 10.8 units), with a cost $12\cdot\rho_p=12\cdot11=132$. Hence, it is more efficient to activate one free subframe than to allocate 12 RBs in class-*p* subframes.
We need to cover 100 full RBs of equivalent capacity, hence we can safely activate 6 free subframes, thus covering $0.85\cdot40+1\cdot10\cdot6=94$ worth of capacity. We are left with 6 RBs. There is still

□

Two issues remain to be solved. The first one is how to determine a suitable value for K. The second one is how to share the RBs among the subframes, given that there exist possibly many ways to do so. We deal with both in the following subsections.

```
1.  split $\Pi$ into $L=\{i\in\Pi:\rho_i/x_i\le\rho_f/x_f\}$, $H=\{i\in\Pi:\rho_i/x_i>\rho_f/x_f\}$
2.  sort the indexes in $L$ and $H$ by increasing PCR, so that
    $i\le j\Leftrightarrow\rho_i/x_i\le\rho_j/x_j$
3.  set $a=0$, $b_i=0\ \ \forall i\in\Pi\cup\Phi$, $k=K$
4.  for i = 1 to $|L|$, do                       // subframes in L first
5.        allocate $b_i=\min\{\lceil K/x_i\rceil,T_i\cdot N\}$ RBs to class i in $L$
6.        set $k=k-b_i\cdot x_i$
7.        if $k\le 0$ then goto end
8.  For i = 1 to $|H|$ do
9.  Set $\beta_i=\min\left(T_i\cdot N,\left\lceil\left(N\cdot x_f-\sum_{j=0}^{i-1}\beta_j\cdot x_j\right)/x_i\right\rceil\right)$
10. if $x_f>\sum_{i\in H}x_i\cdot T_i$ or $\left(P_{base,f}-P_{off}\right)+\rho_f\cdot N\le\sum_{i\in H}\rho_i\cdot\beta_i$ then //check free subframes
11.       activate $d=\min\{\lfloor k/(N\cdot x_f)\rfloor,T_f\}$ free subframes with $N$ RBs each
12.       set $a=a+d$
13.       set $k=k-d\cdot N\cdot x_f$
14.       if $a<T_f$ then        // k/x$_f$ is less than one free subframe
15.             for i = 1 to $|H|$ do
16.             set $\beta_i=\min\left(T_i\cdot N,\left\lceil\left(k\cdot x_f-\sum_{j=0}^{i-1}\beta_j\cdot x_j\right)/x_i\right\rceil\right)$
17.             if $x_f\cdot k>\sum_{i\in H}x_i\cdot T_i\cdot N$ or $\left(P_{base,f}-P_{off}\right)+\rho_f\cdot k\le\sum_{i\in H}\rho_i\cdot\beta_i$ then
18.                   activate one free subframe with $\lceil k/x_f\rceil$ RBs
19.                   set $a=a+1$
20.                   set $k=k-\lceil K/x_f\rceil\cdot x_f$
21.                   goto end
22. for i = 1 to $|H|$, do                              // subframes in H
23.       allocate $b_i=\min\{\lceil k/x_i\rceil,T_i\cdot N\}$ RBs to class i in $H$
24.       Set $k=k-b_i\cdot x_i$
25.       if $k\le 0$ then goto end
26. activate $d=\min\{\lfloor k/(N\cdot x_f)\rfloor,T_f\}$ free subframes with $N$ RBs   // free
    subframes again
27. set $a=a+d$
28. set $k=k-d\cdot N\cdot x_f$
29. if $k\le 0$ then goto end
```

Figure 4-3: Pseudo-code for the throughput-oriented heuristic

```
activate $d = \min\left\{\left\lfloor k/\left(N \cdot x_f\right)\right\rfloor, T_f\right\}$ free subframes with $N$ RBs //free subframes again
set $a = a + d$
set $k = k - d \cdot N \cdot x_f$
if $k \le 0$ then goto end
if $\beta_i = \min\left(T_i \cdot N_i \left\lceil \left(k \cdot x_f - \sum_{j=0}^{i-1} \beta_j \cdot x_j\right) / x_i \right\rceil\right)$ then // k/x_f is less than one free subframe
        activate one free subframe with $\lceil k/x_f \rceil$ RBs
        set $k = k - \lceil K/x_f \rceil \cdot x_f$
        set $k = k - \lceil k/x_f \rceil \cdot x_f$
end:
        while $\Theta = \{i \in \Pi : b_i > 0 \wedge x_i < (-k)\}$ is not empty
                let $j = \arg\max_{i \in \Theta} \{\rho_i\}$
                let $b_j = b_j - 1$
```

4.4.2 Predicting the traffic demand

So far we have assumed that the traffic demand for a superframe is known in advance, it being equal to K fully exploitable RBs. As far as traffic demand is concerned, there are three sources of uncertainties. The first one is that arrivals vary over time, both because flows come and go, *and* because flows are inherently non-homogeneous (e.g., compressed multimedia flows, web browsing, etc.). The second one is that the UE channel status (hence the transmission rate to each UE) varies as well. This may also influence the arrival rates in some case (it is in fact well known that TCP sources adapt to the available bandwidth). The variation in the channel status are exploited by the scheduler, which – broadly speaking – strives to serve the subset of UEs that exhibit the best channel conditions. A final source of uncertainty in a MU-MIMO capable scheduler is that, depending on the channel conditions, two UEs may be paired and served in MU-MIMO in the same RB (or group thereof). This means that the *actual* number of RBs required to carry a given traffic demand depends on whether, and to what extent, MU-MIMO is exploited. Theoretically speaking, if *all* the UEs could be paired in MU-MIMO (while keeping the same rate), then half the RBs would be required to carry out the same job than if they were scheduled in isolation.

This makes it impossible to have absolute certainty on the number of RBs that are required in a future MBSFN superframe. One can only make an informed guess, based on historical records of the relevant quantities, and react – by either reducing or increasing the number of RBs/active frames – to changing conditions. However, the future demand can be *estimated* by considering:

a) the current state of the UE queues (which counts as unsatisfied demand);
b) the percentage of actually allocated RBs in the active subframes in the recent past;
c) the last transmission rate of the UEs;
d) the percentage of MU-MIMO pairings on the allocated RBs in the recent past the retransmission rate;

and assuming that neither the traffic nor the channel conditions are going to change too much in the near future, at least on a cell-wise perspective. The "recent past" in the previous assertion is *the last MBSFN superframe*. In fact, it makes no

sense to trace the allocation conditions too far back, as the further you go back, the less significant those percentages will be.
In estimating the future demand, we only need to determine what is the new number K of required fully-exploitable RBs. The formula is the following:

$$K=\left\lceil\left(\frac{1}{1+\mu}\cdot\sum_{j=1}^{N_{UE}}\left\lceil q_j/R_j\right\rceil+\sum_{i\in\Pi\cup\Phi}A_i\cdot x_i\right)\cdot(1+p)\right\rceil \qquad (4)$$

where A_i is the number of RBs *allocated by the scheduler* to class-*i* subframes of the past superframe, μ is the percentage of RBs where MU-MIMO pairing was activated in the past superframe, q_i is the *current* backlog of UE j and R_j is the most recent sample of the individual rate of UE j (we assume a default value if j has never been served before), N_{UE} is the number of UEs being considered, and p is the retransmission probability (we assume here that the number of double, triple etc. retransmission is negligible). Mark the phrase "allocated by the scheduler", which means those that the scheduler actually *did* exploit for serving traffic, which are less than or equal to those *made available* to the scheduler by the provisioning algorithm for that subframe.

4.4.3 Distributing RBs among active frames

The algorithm outputs a number of RBs for each subframe class, and a number of active free subframes. As already specified, as long as we assume that every RB is exploitable, it is irrelevant whether RBs are shifted from one subframe to another of the same subframe class, since this does not change either the cost or the throughput of a solution.
If b_i RBs are allocated to class-*i* subframes, it is reasonable to split the RBs allocated in each subframe as follows: $\lfloor b_i/N\rfloor$ class-*i* subframes get exactly N RBs one class-*i* subframe gets $b_i \bmod N$ RBs (if greater than zero).
Henceforth, we denote with $n(h)$ the number of RBs that subframe *h* manages as a result of the above partition, and with $c(h)$ the class subframe *h* belongs to.
If we consider that PDU transmission is atomic, it may happen that some RBs go unexploited because a PDU does not fit into the remaining RBs. This cannot happen in the long run (e.g., in a timespan including several superframes, since unsatisfied capacity readily translates into queues building up in (4)). However, we may employ some care to limit this problem within possible. Assume that class *i* is not fully depleted. The number of wasted RBs in the first $\lfloor b_i/N\rfloor$ subframes can be added to a *deficit* counter, i.e. a nonnegative variable d_i, initially null. In the *last* active subframe, the scheduler is allowed to use $\max\{(b_i \bmod N)+d_i, N\}$ RBs. Note that a RB can be considered as "wasted" if and only if there actually is a PDU that should be transmitted, and the latter does not fit into the remaining RBs in the current subframe.
The deficit variable is reset at the end of the superframe.

4.5 Making decisions on a per-TTI basis

So far we have assumed that activation decisions have to be made on a per-superframe basis, and are enforced accordingly at the superframe boundary. Assume that, under the same conditions and power/capacity models as described in Sections 4.2 and 4.3, an architecture mechanism exists that enables an eNodeB to go to sleep at each TTI where a free frame is located, without prior notice. In this case, we may devise a dynamic algorithm, that exploits the *current* channel and backlog conditions to schedule an (in)active subframe. Fresh channel information is surely more reliable than the one from the previous superframe period. Furthermore, such algorithm can look ahead in the future (with a grain of salt, of course), capitalizing on the fact that a future pinned frame will be active nonetheless, and decide to send the eNodeB to sleep just because of that. Finally, as a dynamic algorithm only makes *short-term* decisions (as opposed to the former, which only makes long-term ones), it is reasonable to think that it may base its decision on the *QoS constraints* of the backlogged traffic, instead of just on the required throughput.

Suppose in fact that some UEs have traffic whose deadline is about to expire. Depending on their number, class of service, and performance targets of the network administrator, it may be worth the while to pay for the fixed power offset of activating a (possibly otherwise superfluous) free subframe just for the sake of meeting these UEs' deadlines. On the other hand, if those deadlines are relatively far ahead in the future, and we do not foresee an immediate surge of traffic in the next 3-4 TTIs, it may be better to wait for the next pinned subframe to occur, since it will be large enough to accommodate that amount of real-time traffic without incurring fixed power costs.

We first devise a simpler algorithm, which is oblivious to QoS constraints, and then show how to refine it in order to keep QoS constraints into account. This algorithm should decide whether to *activate* a subframe, i.e. to enable the scheduler to manage that subframe's RBs. If the subframe being decided upon is a *pinned* one, the decision of not activating it means that it will only be used to transmit the required reference signals (broadcast, synchronization, paging, etc.). If, instead, it is a free one, not activating it will put the eNodeB to sleep.

The key parameter of the dynamic algorithm is that of *look-ahead period* T_{la}. The latter is the number of future TTIs where the algorithm both estimates and tries to accommodate the traffic demand, and it is $T_{la} > 1$.

It is clear that the dynamic algorithm must estimate both the traffic demand and the available capacity within the look-ahead period. It therefore maintains the following quantities:

- a running estimate of the *average arrival rate* A^{est}, updated on each TTI t as $A^{est} \leftarrow (1-\alpha_A)\cdot A^{est} + \alpha_A \cdot A(t)$, where $A(t)$ are the arrivals measured at time t and α_A is a dampening factor for the exponential average, $0 \le \alpha_A \le 1$. The higher α_A, the more reactive to sudden changes is the estimate.
- A running estimate of the *average bits per RB for served UEs* C^{est}, updated on each TTI t as $C^{est} \leftarrow (1-\alpha_C)\cdot C^{est} + \alpha_C \cdot C(t)/x_i$, where $C(t)$ is

the quantity measured at time t and α_C is a dampening factor for the exponential average, $0 \le \alpha_C \le 1$, and x_i is the percentage of available per-RB bits in the type of subframe being transmitted at time t.

- A running estimate of the *average ratio of MU-MIMO pairings* μ^{est}, updated on each TTI t as $\mu^{est} \leftarrow (1-\alpha_\mu)\cdot\mu^{est} + \alpha_\mu \cdot \mu(t)$, where $\mu(t)$ is the quantity measured at time t and α_μ is a dampening factor for the exponential average, $0 \le \alpha_\mu \le 1$.

Note that $\alpha_A, \alpha_C, \alpha_\mu$ need not be the same number, as the three above phenomena have different time constants.

In order to estimate the demand for the look-ahead period, we assume that the arrival rate, the number of bits per RB, and the ratio of MU-MIMO pairings will keep *constant* in the future. Obviously, the larger T_{la}, the less reasonable this assumption appears to be. However, as we will see later on in this section, the expected order of magnitude for T_{la} is around 4-5 TTIs, which makes that assumption one we can easily live with. Note that, in any case, decisions are made on a per-TTI basis, hence can be corrected as fast should the surrounding conditions change abruptly.

Under the above hypotheses, the traffic demand in the look-ahead period can be estimated similarly as in (4), i.e.:

$$K = \left\lceil \frac{1}{1+\mu^{est}} \cdot \left\lceil \frac{\left(\sum_{j=1}^{N_{UE}} q_j + A^{est}\cdot T_{la}\right)\cdot(1+p)}{C^{est}} \right\rceil \right\rceil \tag{5}$$

where K is expressed as a number of fully-exploitable RBs. The pseudo-code of the algorithm is shown in Figure 4-4.

```
1.  on each TTI
2.    update $A^{est}$, $C^{est}$, $\mu^{est}$
3.    compute $K$ through (5)
4.    compute $C_{1,T-1} = C^{est} \cdot N \cdot \sum_{i=1}^{T_{la}-1} x_i$
5.    if $K > C_{1,T-1}$ then
6.            activate the current subframe
7.    else
8.            compute $P^{on}$ and $P^{off}$ by executing algorithm 4.4.1
9.            if $P^{on} < P^{off}$ then
10.                   activate the current subframe
```

Figure 4-4. Pseudo-code for the dynamic algorithm

The algorithm can be described in a fairly straightforward way: as long as there is enough capacity in the *rest* of the look-ahead period to sustain the current foreseen demand K, one should allocate the current frame only if that allocation leads to a

smaller-cost solution in terms of power consumption. To this aim, the optimal power consumption for a given demand on a pre-defined period can be computed through the algorithm described in Section 4.4.1. We then compute P^{on}, the (otherwise) power-optimal allocation on the look-ahead period *given that the current subframe is fully exploited*, and P^{off}, the (otherwise) power-optimal allocation *given that the current subframe is not exploited*, and compare them. The solution with the smallest power consumption is then made effective.
Note that, when the above algorithm decides to activate a subframe, the scheduler is enabled to manage the *whole system bandwidth*. In fact, it makes no sense to spare RBs when activating a subframe. Furthermore, note that – when computing P^{on}, algorithm 4.4.1 should take care to compute it as the sum of:

a) the power cost of the virtual schedule made by the scheduler for the current TTI (i.e., not counting future arrivals), and
b) the cost of satisfying (in the remaining part of the look-ahead period) *all* the demand left unsatisfied in the current TTI.

Note that this algorithm cannot slack indefinitely. In fact, if it keeps leaving subframes unused, the condition at line 5 eventually becomes true, which activates the current frame.

4.6 Results

The algorithm presented in section 4.4 has been implemented as a module of the simulator presented in section 3.6.1. The system model is the same illustrated in section 0. In order to evaluate the proposed provisioner we simulate the same scenario with different scheduler:

- PF Legacy
- Max-CI Legacy
- Max-CI++ presented in sec 3.6.2
- QoSMuMIMO presented in chapter 3
- QoSMuMIMO with MBSFN switch.

PF and Max-CI Legacy are the original and well known scheduler without the TMS algorithm and the MCS adaptive allocation. The transmission mode used with these schedulers is Transmit Diversity.
The traffic model simulated is Video On Demand with the same characteristics presented in Table 3-2.
The analyzed metrics are:

- Cell throughput
- User Average Delay
- Depleted Power

4.6.1 Simulated scenarios

In order to evaluate the performance of the provisioner in terms of depleted power we simulate some scenarios with increasing network load. In Table 4-1 we report the number of user in the cell in function of the offered load. In the first scenario that we evaluate the provisioner algorithm is executed every 80 TTI.

Offered Load (Mbit/s)	Number of UE
0,5	2
1	4
1,5	6
2,5	10
5	20
8	32
10	40
20	80

Table 4-1: Number of UE for each offered load

In Figure 4-5 the depleted power with increasing network load is shown. We note that without the proposed provisioner all the schedulers would offer similar performance, while QosMuMimo jointly with the proposed provisioner provide better performance in terms of depleted power. In particular with low traffic load the energy consumed is 30% less than the other schedulers without the provisioner. The benefits provided by the provisioner decrease linearly with the offered load. This is due to the characteristics of the proposed provisioner: with low load the provisioner allocates a few packets in the *pinned* subframes allowing eNodeB to enable sleep mode in the *MBSFN* subframes. With increasing load an higher number of MBSFN frames will be activated by the provisioner in order to transmit backlogged data, resulting in an increased energy consumption.

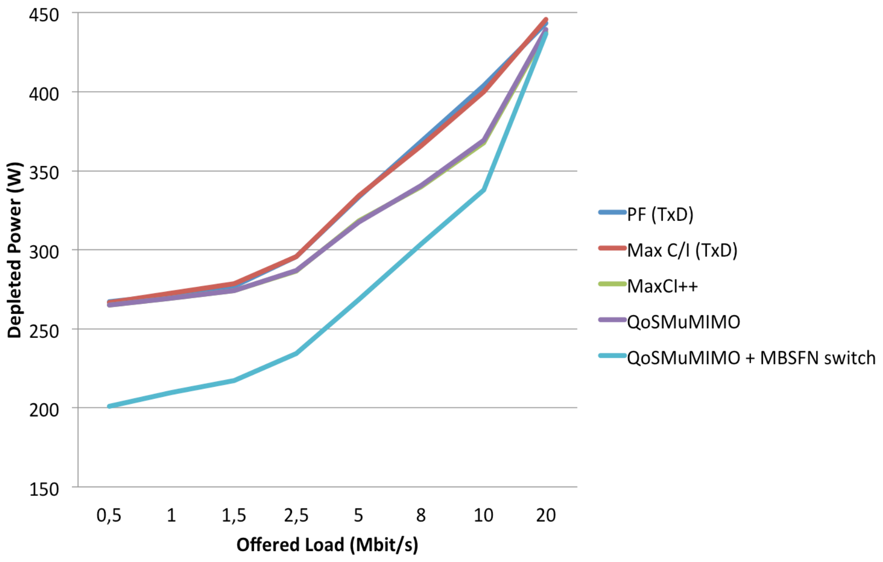


Figure 4-5 Depleted power with increasing network load

In Figure 4-6 we report the average delay when the network load increases. We can see that with low cell load QosMuMimo + proposed provisioner add a small amount of delay respect the other scheduler that do not use the provisioner. This can be explained with one of the following reasons:

1. with low cell load many frames are put in sleep mode, consequently backlogged data has to wait the pinned frames (or active frames) to be transmit
2. the algorithm that predicts the traffic demand (described in section 0) reports an underestimate prediction of the future traffic demand causing an increasing delay for the data backlogged for the previous reason.

When cell is overloaded (cell load higher than 20 Mbit/s) the proposed provisioner does not affect the result as described above.

MaxCI and PF provide lower performance than the other scheduler because they do not exploit the TMS algorithm.

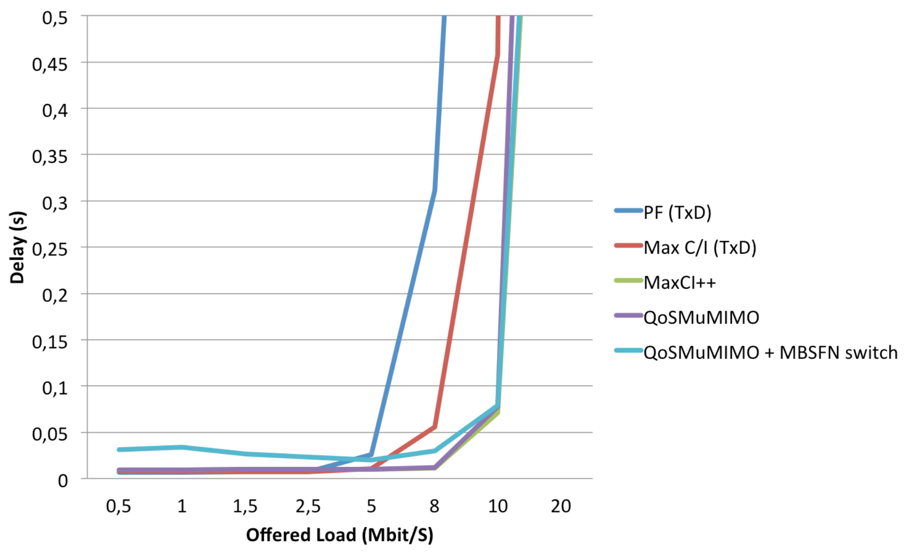


Figure 4-6 Average delay with increasing network load

The same behavior can be seen in Figure 4-7 where the cell throughput with 2MB/s of offered load is reported. MaxCI++ and QosMuMimo are able to transmit all the incoming traffic while MaxCI and PF are not efficient enough.

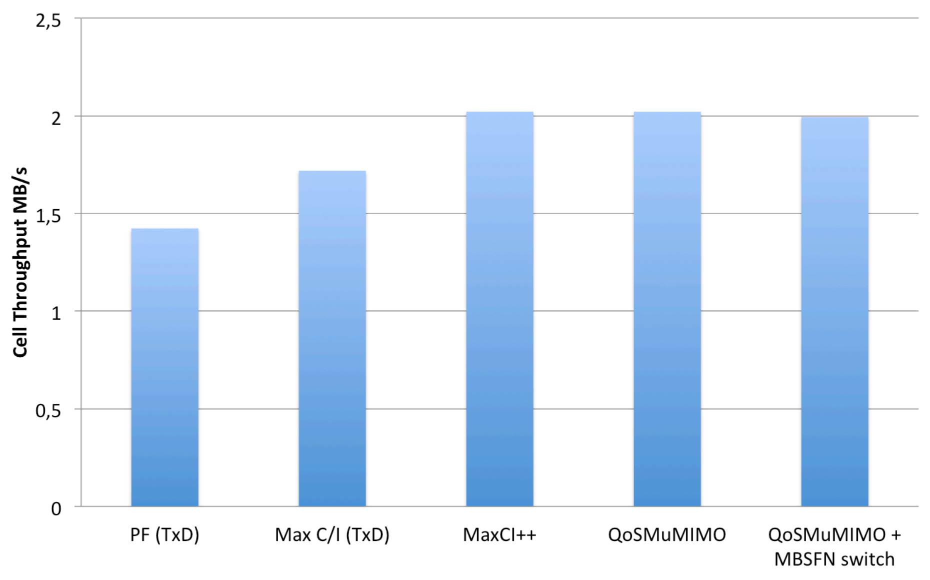


Figure 4-7 Cell Throughput with 2 MB/s of network load

The next two graphs are obtained simulating the QosMuMimo + MBSFN switch scheduler, varying the activation time window of the provisioner.
In Figure 4-8 we can note that the depleted power is not affected by the activation time window of the provisioner, in fact the results do not change with different values of the windows size and the cell load.

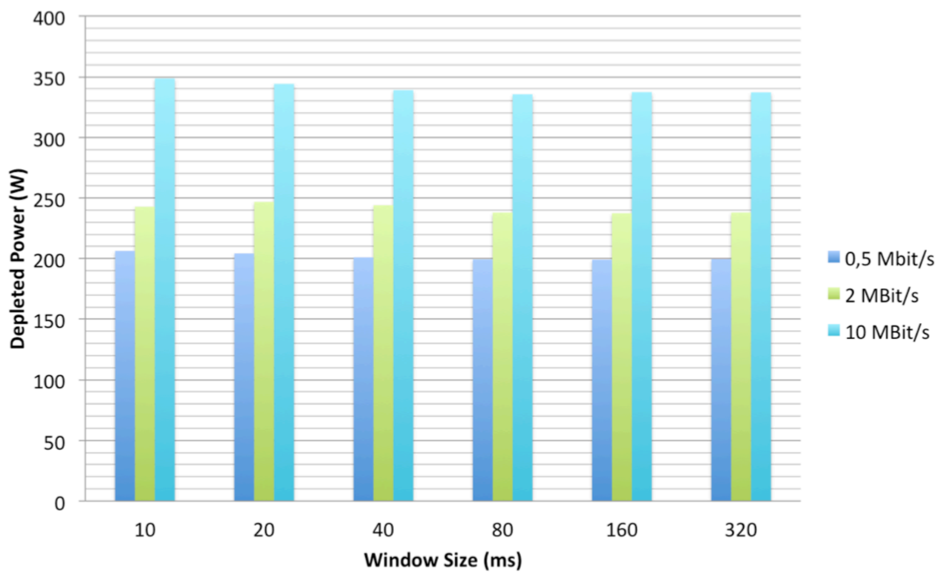


Figure 4-8 Depleted power with increasing activation time of the provisioner

Finally in Figure 4-9 is reported the average delay with different cell load and different window size. We can note that the average delay increases proportionally with the window size. This is the side effect of the algorithm that predict the traffic demand because the higher the window size, the bigger the estimation error. We can also note that when window size is equal to 10 the average delay is similar to the case where the provisioner is disabled (Figure 4-6).

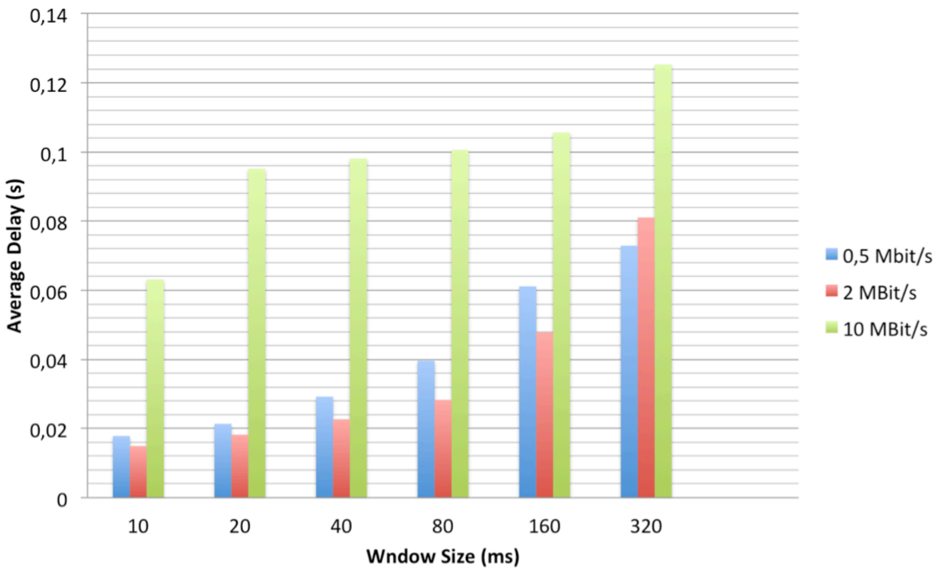


Figure 4-9 Average delay with increasing activation time of the provisioner

5 CONCLUSIONS

In this work we addressed several aspects of the downlink-scheduling problem in LTE Advanced MU-MIMO systems, proposing a modular scheduling framework where transmission mode selection, frequency domain allocation, QoS demand, fairness among users, and energy efficiency are managed all together, respecting the constraints imposed by the LTE standard.
The problem of transmission mode selection and frequency domain allocation exploiting multi-user diversity has been tackled formalizing, in the first place, a throughput-optimal mathematical model, which we proved to be NP-hard. Given the hardness of the problem, in the second place we designed an heuristic polynomial-time algorithms for transmission mode selection.
To support the QoS requirements of different traffics, we developed an innovative scheduling discipline able to properly manage *delay-sensitive* traffics, ensuring also fairness and spectral efficiency. The simulations performed have pointed out that the proposed solution guarantees the fulfillment of deadlines for *real-time* applications and ensures a good fairness to background users, exhibiting high throughput results at the same time.
Finally we propose a provisioner that exploits the DTX technique of LTE in order to minimize the energy consumption of an eNodeB with low or no impact in terms of throughput and delay performance. The results show that the usage of the proposed provisioner can provide a decreased energy consumption up to 30%.

6 REFERENCES

1. *Third Generation Partnership Project.* http://www.3gpp.org/
2. 3GPP. *Long Term Evolution.* http://www.3gpp.org/LTE
3. 3GPP. *LTE Advanced.* http://www.3gpp.org/LTE-Advanced
4. *International Telecommunication Union.* http://www.itu.int
5. TS36.211. Physical Channels and Modulation. Technical report, 3rd Generation Partnership Project, 2011
6. Tetsushi Abe. Further Enhancements for LTE-Advanced. NTT DoCoMo, 2010.
7. OMNeT++. http://www.omnetpp.org
8. Harri Holma and Antti Toskala. *LTE for UMTS OFDMA and SC-FDMA Based Radio Access.* John Wiley & Sons, 2009
9. TS36.323. Packet Data Convergence Protocol (PDCP) specification. Technical report, 3rd Generation Partnership Project, 2010.
10. TS36.331. Radio Resource Control (RRC) Protocol specification. Technical report, 3rd Generation Partnership Project, 2010.
11. TS36.322. Radio Link Control (RLC) protocol specification. Tech- nical report, 3rd Generation Partnership Project, 2010.
12. TS36.321. Medium Access Control (MAC) protocol specification. Technical report, 3rd Generation Partnership Project, 2010.
13. TS36.201. LTE physical layer general description. Technical report, 3rd Generation Partnership Project, 2010.
14. 3GPP - TS 36.300 *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description*; Stage 2
15. 3GPP - TS 36.913 *Requirements for further advancements for Evolved Universal Terrestrial Radio Access (E-UTRA) (LTE-Advanced)*
16. S.-B. Lee, S. Choudhury, a. Khoshnevis, S. Xu and S. Lu. "*Downlink MIMO with Frequency-Domain Packet Scheduling for 3GPP LTE*",. IEEE INFOCOM 2009, pp. 1269–1277, Apr. 2009
17. J. Joung and Y. H. Lee, "*Evaluation for Various Resource Allocation Methods for Multiuser-MIMO OFDMA Systems*". IEEE PIMRC 2007, pp. 1-5, sep 2007.
18. N. Wei, A. Pokhariyal, T. Sørensen, T. Kolding and P. Mogensen, "*Performance of Spatial Division Multiplexing MIMO with Frequency Domain Packet Scheduling: From Theory to Practice*", IEEE JSAC, pp 890-900, aug 2008.
19. M. Nicolaou, A. Doufexi, S. Armour and Y. Sun, "*Scheduling Techniques for Improving Call Capacity for VoIP Traffic in MIMO-OFDMA Networks*". 2009 IEEE *70*th Vehicular Technology Conference Fall, pp. 1-5, Sep. 2009.
20. W. Huang and R. Song, "*A novel QoS-guaranteed proportional fairness with joint space-time-frequency scheduling*", 15th Asia-Pacific Conference on Communications, pages 135–138, Oct. 2009
21. T. Bonald "*A Score-Based Opportunistic Scheduler for Fading Radio Channels*", Proc European Wireless, page 2, 2004

22. O. Souihli and T.Ohtsuki, "*Joint Feedback and Scheduling Scheme for Service-Differentiated Multiuser MIMO Systems*". 2009 IEEE 70th Vehicular Technology Conference Fall, sep 2009.
23. K. C. Beh, A. Doufexi and S. Armour, "*On the performance of SU-MIMO and MU-MIMO in 3GPP LTE downlink*", IEEE PIMRC 2009, pp. 1482-1486, Sep. 2009.
24. J. Su, T.Pei, Z. He and Z. Luo, "*Multiuser Pairing Scheduling Considering QoS in Virtual MIMO System*", 2010 International Conference on Communications and Mobile Computing (CMC), 2010.
25. J. Duplicy *et al.*, "*MU-MIMO in LTE Systems*", EURASIP JWCN, Vol. 2011, Article ID 496763, 13 pages
26. *C++ Programming Language.* http://www.cplusplus.com/.
27. Simulcraft. The OMNEST Network Simulation Framework. http: //www.omnest.com/
28. Moving Picture Experts Group. MPEG-4. http://mpeg. chiariglione.org/standards/mpeg-4/mpeg-4.htm
29. Wimax Forum. WiMAX System Evaluation Methodology. Forum American Bar Association, pages 1–209, 2008
30. TS23.203. Policy and charging control architecture. Technical report, 3rd Generation Partnership Project, 2011
31. ITU, "G.113: Transmission impairments due to speech processing," 2001
32. Jalali, R. Padovani, and R. Pankaj, "Data throughput of CDMA-HDR a high efficiency-high data rate personal communication wireless system," Proc. IEEE VTC'00, Tokio, Japan, May 15–18, 2000, pp. 1854–1858.
33. M. Andreozzi, D. Migliorini, G. Pagano, G. Stea, D. Sabella, M. Caretti, R. Fantini, "LTE Scheduling", Patent PCT/EP2011/053225, filed 03/03/2011
34. M. Caretti, C. Cicconetti, D. Franceschini, D. Migliorini, L. Lenzini, E. Mingozzi, R. Rossi, D. Sabella, 'Efficient Downlink Scheduling with Power Boosting in Mobile IEEE 802.16 Networks', Computer Networks, Vol. 55, No. 8, pp. 1672-1683, June 2011.
35. C. Cicconetti, L. Lenzini, D. Migliorini, E. Mingozzi, 'Power-aware Opportunistic Downlink Scheduling in IEEE 802.16 Wireless Networks', Computer Communications, Vol. 34, No. 17, pp. 2026-2035, November 2011.
36. D. Migliorini, E. Mingozzi, C. Vallati, 'Performance evaluation of H.264/SVC video streaming over mobile WiMAX', Computer Networks, Vol. 55, No. 15, pp. 3578-3591, October 2011.
37. C. Cicconetti, L. Lenzini, D. Migliorini, E. Mingozzi. "Power-Aware Opportunistic Downlink Scheduling In IEEE 802.16 Wireless Networks". European Wireless 2010, Lucca, Italy. 12-16 April 2010.
38. M. Andreozzi, D. Migliorini, G. Stea, C. Vallati. "Ns2Voip++, an enhanced module for VoIP simulations." SIMUTools 2010, Torremolinos, Malaga, Spain, 15-16 March 2010
39. D. Migliorini, E. Mingozzi, C. Vallati. "QoE-oriented performance evaluation of video streaming over WiMAX". Proceedings of the Eighth International Conference on Wired/Wireless Internet Communications (WWIC 2010), Luleå, Sweden, June 1-3, 2010.
40. M. Nicolaou, A. Doufexi, S. Armour and Y. Sun, "Scheduling Techniques for Improving Call Capacity for VoIP Traffic in MIMO-OFDMA Networks".

2009 IEEE *70*th Vehicular Technology Conference Fall, pp. 1-5, Sep. 2009.

41. P. Frenger, P. Moberg, J. Malmodin, Y. Jading, I. Gòdor "Reducing Energy Consumption in LTE with Cell DTX", VTC Spring, 2011
42. Generoso Pagano, "*Design and Evaluation of a Multi-User MIMO Scheduler Supporting Quality of Service for LTE Advanced Networks*" Master Degree Thesis 2011