Abstract

Designing scheduling algorithms that work in synergy with TCP is a challenging problem. In this thesis, we focus on Max Weight (MW) schedulers and show that contrary to its performance with inelastic traffic, MW schedulers may not perform well in LTE networks in the presence of TCP traffic. We then design two new schedulers which both are based on MW. The first proposed scheduler, called Q-MW, is tailored specifically to TCP dynamics by giving higher priority to TCP flows whose queue at the base station is very small to encourage them to send more data at a faster rate. The second proposed scheduler is named H-MW in which some of the subcarriers with low channel quality are left unassigned to improve the total MW scheduler throughput. In the last part of the thesis, we consider how LTE handovers affect the performance of scheduling algorithms under TCP traffic.
Acknowledgements

Firstly, I want to express my sincere thanks to my supervisor, Dr. Majid Ghaderi, for his supervision and support over the past two and a half years. His knowledge and recommendations on a large variety of topics from how to give effective presentations to how to improve paper and thesis have been invaluable and much appreciated.

A genuine thank you goes to my graduate committee, which consisted of Dr. Mea Wang and Dr. Diwakar Krishnamurthy, for their feedback and constructive comments, which has improved the final version of this thesis.

I would like to give a big shout out of gratitude to my husband, Abbas, for his unconditional love and support. He was by my side throughout this challenging period, and encouraged me whenever I wanted to give up. He was the one who helped me to solve the problems when no one did and I can never thank him enough.

A special thank also goes to my family for their tremendous support and motivation they gave to me during this journey. Words cannot express how grateful I am to my lovely mom and dad for their sacrifices. Thank you to my brothers and my in-laws for always believing in me. Your prayer for me was what sustained me this far.

Finally, and most importantly, I thank God, for letting me through all the difficulties. I have experienced Your guidance day by day. Thank you.
# Table of Contents

**Abstract** ......................................................... i

**Acknowledgements** ........................................... ii

**Table of Contents** ............................................ iii

**List of Tables** ................................................ vi

**List of Figures** ................................................ vii

**List of Symbols** .............................................. ix

1 **Introduction** .................................................. 1

1.1 LTE Networks .................................................. 2

1.2 Packet Scheduling ............................................. 2

1.3 Problem Statement ............................................. 3

1.3.1 Performance of Existing Schedulers ......................... 3

1.3.2 Proposed Schedulers ....................................... 4

1.3.3 Handover Effect on the Performance of Scheduling Algorithms 6

1.4 Thesis Objectives and Contributions ......................... 8

1.5 Thesis Outline ............................................... 9

2 **Related Work** ................................................ 10

2.1 Channel-Unaware Scheduling Strategies ...................... 10

2.1.1 Round Robin Scheduler .................................. 11

2.1.2 Blind Equal Throughput Scheduler ...................... 11

2.2 Channel-Aware/QoS-Unaware Scheduling Strategies ........... 12

2.2.1 Maximum Throughput Scheduler ............................ 12

2.2.2 Proportional Fair Scheduler ............................... 12

2.2.3 Throughput to Average Scheduler ......................... 13

2.2.4 Joint Time and Frequency Domain Scheduler .............. 14

2.2.5 Delay Sensitive Scheduler ................................ 14

2.2.6 Buffer-Aware Scheduler .................................. 14

2.2.7 TCP-Aware Scheduler ..................................... 15

2.3 Channel-Aware/QoS-Aware Scheduling Strategies ............. 16

2.3.1 Token Bank Fair Queue Scheduler ....................... 16

2.3.2 Priority Set Scheduler ................................... 18

3 **LTE Primer** .................................................. 20

3.1 Introduction .................................................. 20

3.2 LTE System Architecture .................................... 22

3.2.1 Serving Gateway .......................................... 22

3.2.2 Packet Data Network Gateway ............................ 23

3.2.3 Mobile Management Entity ............................... 24

3.2.4 Home Subscriber Server/Policy Control and Charging Rules Function 25

3.3 LTE Air Interface and Radio Network ....................... 25

3.3.1 LTE Radio Protocol Architecture ......................... 25

3.3.2 LTE Physical Layer for Downlink Transmission: OFDMA 27

3.3.3 LTE Physical Layer for Uplink Transmission: SC-FDMA 29

3.3.4 LTE Resource Allocation .................................. 30
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.5</td>
<td>The LTE Channel Model in Downlink Direction</td>
<td>31</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Downlink Management Channels</td>
<td>32</td>
</tr>
<tr>
<td>3.4</td>
<td>Downlink Scheduling</td>
<td>33</td>
</tr>
<tr>
<td>3.5</td>
<td>RRM Procedures in LTE</td>
<td>37</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Adaptive Modulation and Coding (AMC)</td>
<td>37</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Hybrid Automatic Repeat Request (HARQ)</td>
<td>38</td>
</tr>
<tr>
<td>4</td>
<td>LTE Simulation Tools</td>
<td>40</td>
</tr>
<tr>
<td>4.1</td>
<td>Simulation in Networking Area</td>
<td>40</td>
</tr>
<tr>
<td>4.2</td>
<td>ns-3 Simulator</td>
<td>42</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Modular ns-3</td>
<td>43</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Helper/Container APIs</td>
<td>44</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Logging and Tracing in ns-3</td>
<td>45</td>
</tr>
<tr>
<td>4.3</td>
<td>LTE Package of ns-3 Simulator</td>
<td>45</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Channel and Propagation</td>
<td>47</td>
</tr>
<tr>
<td>4.3.2</td>
<td>PHY Model</td>
<td>48</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Hybrid ARQ</td>
<td>48</td>
</tr>
<tr>
<td>4.3.4</td>
<td>MAC Layer Model</td>
<td>49</td>
</tr>
<tr>
<td>4.4</td>
<td>Implementing a Simple LTE/EPC Network</td>
<td>50</td>
</tr>
<tr>
<td>5</td>
<td>TCP-Friendly Schedulers</td>
<td>57</td>
</tr>
<tr>
<td>5.1</td>
<td>Implementing a New Scheduler in ns-3</td>
<td>57</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Accessing and Updating Queue Size</td>
<td>59</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Accessing CQI</td>
<td>60</td>
</tr>
<tr>
<td>5.2</td>
<td>Max Weight Schedulers</td>
<td>60</td>
</tr>
<tr>
<td>5.3</td>
<td>Proposed Scheduling Algorithms</td>
<td>62</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Q-MW Scheduler</td>
<td>62</td>
</tr>
<tr>
<td>5.3.2</td>
<td>H-MW Scheduler</td>
<td>63</td>
</tr>
<tr>
<td>6</td>
<td>Results and Discussion</td>
<td>67</td>
</tr>
<tr>
<td>6.1</td>
<td>Single-Cell Scenario</td>
<td>67</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Network Topology</td>
<td>67</td>
</tr>
<tr>
<td>6.1.2</td>
<td>System Parameters</td>
<td>68</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Fairness Metrics</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>Multi-Cell Scenario</td>
<td>69</td>
</tr>
<tr>
<td>6.2.1</td>
<td>System Parameters</td>
<td>70</td>
</tr>
<tr>
<td>6.3</td>
<td>System Parameters definition</td>
<td>70</td>
</tr>
<tr>
<td>6.4</td>
<td>TCP Throughput with Existing Schedulers</td>
<td>71</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Experimental Design</td>
<td>72</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Long-Lived Saturated Flows</td>
<td>72</td>
</tr>
<tr>
<td>6.4.3</td>
<td>Short-Lived TCP Flows</td>
<td>82</td>
</tr>
<tr>
<td>6.5</td>
<td>TCP Throughput with Proposed Schedulers</td>
<td>83</td>
</tr>
<tr>
<td>6.5.1</td>
<td>Q-MW Performance Evaluation</td>
<td>83</td>
</tr>
<tr>
<td>6.5.2</td>
<td>H-MW Performance Evaluation</td>
<td>85</td>
</tr>
<tr>
<td>6.6</td>
<td>TCP Throughput with Handover</td>
<td>87</td>
</tr>
<tr>
<td>6.6.1</td>
<td>Single-Cell Network</td>
<td>88</td>
</tr>
<tr>
<td>6.6.2</td>
<td>Multi-Cell Network</td>
<td>88</td>
</tr>
<tr>
<td>6.6.3</td>
<td>Discussion</td>
<td>92</td>
</tr>
</tbody>
</table>
List of Tables

3.1 LTE performance requirements .................................................. 21
3.2 Defined bandwidths for LTE .................................................. 28
3.3 Resource allocation RBG size vs. downlink system bandwidth .............. 37

6.1 System parameters .......................................................... 68
6.2 System parameters .......................................................... 70
6.3 Default configuration ........................................................ 72
6.4 Average buffer occupancy .................................................... 92
List of Figures and Illustrations

1.1 High-level LTE network architecture ........................................ 2
2.1 Downlink scheduling of TBFQ scheduler ................................... 17
3.1 The EPS network elements ...................................................... 22
3.2 LTE control and user plane architectures .................................. 26
3.3 Protocol layer architecture ..................................................... 27
3.4 LTE resource blocks in time and frequency domains .................. 30
3.5 LTE downlink channel structure .............................................. 31

4.1 ns-3 software organization ..................................................... 44
4.2 LENA model overview ......................................................... 46
4.3 LTE subframe division .......................................................... 49

5.1 Mac scheduler interface ........................................................ 58
5.2 Heuristic Scheduler .............................................................. 65

6.1 LTE-EPC network topology simulated in ns-3 ............................ 67
6.2 Effect of the number of UEs on the total throughput of different schedulers for long-lived TCP and UDP flows .................................................... 74
6.3 Proportional fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF achieves the highest fairness. For small number of UEs, the indices are very close. In all experiments, MW achieves a poor fairness index ........................................................................ 75
6.4 Jain’s fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF and RR achieve the highest fairness while the other two schedulers have yield much lower metrics. In all experiments, MW achieves a poor fairness index and its performance deteriorates as the number of UEs increases ........................................................................ 76
6.5 Effect of packet size on the performance of different schedulers for long-lived TCP and UDP flows. Regardless of the packet size, MW achieves the highest throughput ........................................................................ 78
6.6 Effect of Internet delay on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput ........................................................................ 79
6.7 Effect of buffer size on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput in all experiments. .................................................... 81
6.8 Moving averages of buffer occupancy for UE 1, 2, 3, and 4 with MW for 100 packets buffer size and 50 ms Internet delay. UE 1 has a very low queue occupancy for a sustained period of time ........................................................................ 82
6.9 Response time comparison for short-lived TCP flows. Overall, there is no significant difference between the response times of different schedulers ........................................................................ 84
6.10 Throughput gain of MW over PF for different combinations of buffer size and Internet delay. For some combinations, PF outperforms MW ........................................................................ 85
6.11 Throughput gain of Q-MW over PF and MW for different combinations of
buffer size and Internet delay. .................................................. 86
6.12 Heuristic scheduler ............................................................... 87
6.13 Effect of Internet delay in a single-cell network: MW achieves higher through-
put in most experiments. .......................................................... 89
6.14 Effect of buffer size in a single-cell network: MW achieves higher throughput
in most experiments. ............................................................... 90
6.15 Effect of Internet delay in a multi-cell network: PF outperforms MW in most
experiments. ............................................................ 91
6.16 Effect of buffer size a multi-cell network: PF outperforms MW in most exper-
iments. ............................................................... 93
## List of Symbols, Abbreviations and Nomenclature

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>3GPP</td>
<td>3rd Generation Partnership Project</td>
</tr>
<tr>
<td>ACK</td>
<td>Acknowledgment</td>
</tr>
<tr>
<td>AMC</td>
<td>Adaptive Modulation and Coding</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>ARQ</td>
<td>Automatic Repeat Request</td>
</tr>
<tr>
<td>BER</td>
<td>Bit Error Rate</td>
</tr>
<tr>
<td>BLER</td>
<td>Block Error Rate</td>
</tr>
<tr>
<td>CQI</td>
<td>Channel Quality Generator</td>
</tr>
<tr>
<td>CWND</td>
<td>Congestion Window</td>
</tr>
<tr>
<td>DCI</td>
<td>Data Control Indication</td>
</tr>
<tr>
<td>DL</td>
<td>Downlink</td>
</tr>
<tr>
<td>ECC</td>
<td>Error Correction Code</td>
</tr>
<tr>
<td>eNB</td>
<td>Evolved NodeB</td>
</tr>
<tr>
<td>EPC</td>
<td>Evolved Packet Core</td>
</tr>
<tr>
<td>E-UTRAN</td>
<td>Evolved UMTS Terrestrial Radio Access Network</td>
</tr>
<tr>
<td>EV-DO</td>
<td>Evolved Data Optimized</td>
</tr>
<tr>
<td>GNU</td>
<td>General Public License</td>
</tr>
<tr>
<td>GSM</td>
<td>Global System for Mobile Communications</td>
</tr>
<tr>
<td>GTP</td>
<td>GPRS Tunneling Protocol</td>
</tr>
<tr>
<td>HTTP</td>
<td>Hyper Text Transfer Protocol</td>
</tr>
<tr>
<td>IFFT</td>
<td>Inverse Fast Fourier Transformation</td>
</tr>
<tr>
<td>IMS</td>
<td>IP Multimedia Subsystem</td>
</tr>
<tr>
<td>LENA</td>
<td>(ns)-3 Network Simulator</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>LTE</td>
<td>Long Term Evolution Standard</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>MCS</td>
<td>Modulation and Coding Scheme</td>
</tr>
<tr>
<td>MIMO</td>
<td>multiple-input Multiple-output</td>
</tr>
<tr>
<td>MW</td>
<td>Max Weight Scheduling Algorithms</td>
</tr>
<tr>
<td>NAS</td>
<td>Non-access Stratum</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency Division Multiplexing</td>
</tr>
<tr>
<td>OFDMA</td>
<td>Orthogonal Frequency Division Multiple Access</td>
</tr>
<tr>
<td>PAPR</td>
<td>high Peak to Average Power Ratio</td>
</tr>
<tr>
<td>PDCP</td>
<td>Packet Data Convergence Protocol</td>
</tr>
<tr>
<td>PDN</td>
<td>Packet Data Network</td>
</tr>
<tr>
<td>PDU</td>
<td>Protocol Data Unit</td>
</tr>
<tr>
<td>PF</td>
<td>Proportional Fair Scheduling Algorithm</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical Layer</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Assurance</td>
</tr>
<tr>
<td>QAM</td>
<td>Quadrature Amplitude Modulation</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>QPSK</td>
<td>Quadrature Phase Shift Keying</td>
</tr>
<tr>
<td>RB</td>
<td>Resource Block</td>
</tr>
<tr>
<td>RBG</td>
<td>Resource Block Group</td>
</tr>
<tr>
<td>RLC</td>
<td>Radio Link Control</td>
</tr>
<tr>
<td>RR</td>
<td>Round Robin Scheduling Algorithm</td>
</tr>
<tr>
<td>RRC</td>
<td>Radio Resource Control</td>
</tr>
<tr>
<td>RRM</td>
<td>Radio Resource Management</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip Time</td>
</tr>
<tr>
<td>SC-FDMA</td>
<td>Single-Carrier Frequency Division Multiple Access</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>SDU</td>
<td>Service Data Unit</td>
</tr>
<tr>
<td>S-GW</td>
<td>Serving Gateway</td>
</tr>
<tr>
<td>SINR</td>
<td>Signal to Noise Ratio</td>
</tr>
<tr>
<td>SISO</td>
<td>Single-input Single-output</td>
</tr>
<tr>
<td>SRS</td>
<td>Sounding Reference Signal</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>TM</td>
<td>Transparent Mode</td>
</tr>
<tr>
<td>TTI</td>
<td>Transmission Time Interval</td>
</tr>
<tr>
<td>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>UE</td>
<td>User Equipment</td>
</tr>
<tr>
<td>UL</td>
<td>Uplink</td>
</tr>
<tr>
<td>UM</td>
<td>Unacknowledged Mode</td>
</tr>
<tr>
<td>UMTS</td>
<td>Universal Mobile Telecommunication System</td>
</tr>
<tr>
<td>VOIP</td>
<td>Voice over IP</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 LTE Networks

Widespread adoption of different forms of wireless devices leads to the demand for constant connectivity to the Internet. In fact, an overwhelming increase in mobile data traffic, fueled by millions of applications of mobile devices such as smartphones and tablets, and also various wireless connections between objects, requires wireless mobile networks to be consistent and to enable users to download more content, more quickly. This, in turn, results in an ever-growing rise in the demand for wireless spectrum and makes the wireless industry strive to find better ways to address data demands. Cellular networks are one of the most promising technologies to cope with the exponential demand for wireless networks.

To meet the growing demand for cellular services, cellular operators around the world are deploying 4G networks based on the Long Term Evolution (LTE) standard. In summary, the promise of LTE is to provide a high data rate and low latency by providing a packet-optimized wireless access and core network. The high-level architecture of an LTE network is depicted in Fig. 1.1. As can be seen in this figure, LTE is made by a core network, called the “Evolved Packet Core” (EPC), and a radio access network, called the “Evolved Universal Terrestrial Radio Access Network” (E-UTRAN). LTE evolved from the Universal Mobile Telecommunication System (UMTS) where to address UMTS limitations, both radio network and the core network are redesigned. As an example, to overcome the effects of multipath fading available in UMTS, a completely different air interface is provided for LTE. Specifically, the underlying physical layer technology in LTE is Orthogonal Frequency Division Multiple Access (OFDMA) in which the radio frequency is divided into many orthogonal subcarriers.
1.2 Packet Scheduling

Allocation of radio resources to different users is the responsibility of packet scheduling mechanism which works at the radio base station. In case of LTE networks, a multi-user scheduler at the evolved NodeB (eNB) assigns subsets of subcarriers to individual users which allows for flexible bandwidth sharing in the system. To efficiently utilize limited radio resources, wireless schedulers generally take into consideration varying user demands and fluctuating wireless channels, and give a higher priority to users with better channel conditions and/or higher bandwidth requirements. In an LTE network, users regularly report their channel quality indicator (CQI) for each subcarrier to the eNB. Moreover, each user has a dedicated queue at its corresponding eNB to buffer packets that arrive at the eNB for transmission to the user. An LTE scheduler may use the queue size as an indication of a user bandwidth requirement, in conjunction with CQIs, when making scheduling decisions.

There is a large body of work on developing wireless scheduling algorithms that can leverage wireless channel fluctuations [16]. An important class of such algorithms is the Max Weight (MW) scheduling algorithms [61]. In its general form, this class of algorithms prioritizes users with the largest weight for scheduling where the weight is a function of the user queue and channel conditions. There are different approaches for assigning scheduling weights to users [11]. A user’s weight can be simply set to its queue size, its maximum achievable rate, or the product of its queue size and its maximum achievable rate. A desirable property of such algorithms is that they have been shown to be throughput-optimal for a
diverse set of traffic conditions. Specifically, if any algorithm can stabilize user queues, this one can. However, a key premise for this property is that the traffic source rate is de-coupled from the wireless channel and scheduler decisions. This is reasonable for UDP traffic, but the TCP traffic source rate is strongly correlated to the channel. Another popular scheduler that is widely deployed in today’s networks is the proportional fair (PF) scheduler, which seeks to schedule users so as to maximize the proportional fairness metric \[30\]. However, the PF scheduler also de-couples the traffic source rate from the channel.

1.3 Problem Statement

The general goal of this thesis is to study how different LTE scheduling algorithms affect TCP throughput. To this end, we investigate three specific problems in the following subsections.

1.3.1 Performance of Existing Schedulers

There is extensive work on generalizing MW (e.g., to multi-carrier systems \[12\]) and characterizing its performance in terms of delay, throughput and fairness (e.g., see \[36\]). However, most studies consider only inelastic traffic (e.g., UDP). In contrast, a majority of today’s traffic is carried via HTTP, which in turn utilizes TCP. The main reason for this is that HTTP traffic is well understood and can pass through middleboxes, which may block other traffic (e.g., UDP). TCP, unlike UDP, dynamically adjusts its source rate in response to the perceived throughput, which in turn is decided by the scheduler. For example, in conditions where throughput reduces, the TCP source could time-out and drastically reduce its rate, which is undesirable.

While it has been shown that MW schedulers, in conjunction with idealized rate-based congestion controllers, can indeed perform optimally \[21\], in practice, TCP congestion controllers are loss-based and involve several artifacts such as time-outs, which are not accounted for in these works. As such, our goal in the first part of the thesis is to study the performance
of the MW/PF class of schedulers for TCP with simulations that closely mimic real world constraints as well as channel traces to identify any potential shortcomings.

To achieve this goal, we implement several variations of MW (see Chapter 3 in ns-3). We use the LTE model in ns-3 to create an end-to-end LTE network, which has all of the major elements of a real LTE system including the Evolved Packet Core (EPC). Moreover, we use real channel traces in our evaluation. The LTE model in ns-3 provides a detailed implementation of various aspects of the LTE standard [50] such as the OFDMA, hybrid ARQ, adaptive modulation and coding, and handover management.

We study the performance of our implementations of MW in a wide range of network conditions and compare them with other schedulers built-in ns-3 such as PF, Round Robin (RR) and Frequency-Domain Maximum Throughput (FDMT). Our results show that MW may not perform well in some network scenarios, and, in fact, achieve a lower throughput than that of other schedulers such as PF. Particularly, we observe that when there are only a few users in a cell (which is common with increasingly smaller cell sizes of LTE [59]), the throughput of MW drops considerably compared to when there are many users in the cell.

1.3.2 Proposed Schedulers

In the second part of the thesis, we propose two new schedulers, called Q-MW and H-MW. Q-MW scheduler is a TCP-friendly scheduling algorithm designed to remedy the problem observed in the first part of the thesis (Subsection 1.3.1). In this scheduler which is a modified version of MW, those users whose packet queues at the eNB node are below a certain threshold are given higher priority, regardless of their channel and queue conditions. The goal is to speed-up the ACK generation rate from the user and hence encourage TCP to increase its sending rate (following the additive increase mechanism of TCP) to prevent the user queue at the eNB from being depleted. When there are only a few users in a cell, if some of them have no data in their queues, then MW has limited choices in selecting users for transmission. This will negatively affect the throughput achieved by MW. Notice that this
problem is specific to elastic TCP traffic, but it does not happen with inelastic UDP traffic.
With inelastic traffic, in our simulations, user queues always remain saturated. Therefore,
Q-MW has a queue threshold parameter that needs to be tuned properly for a given network
scenario. Our simulation results across a wide range of network configurations, in terms
of queue size at the base station and round-trip delay, show that the proposed Q-MW can
outperform the traditional MW and PF in terms of the achieved throughput as summarized
below:

- With dynamic tuning, by 10% and 11% on average and up to 37% and 64%,
  respectively.

- With static tuning, by 7% and 6% on average and up to 29% and 64%, re-
  spectively.

It should be noted that, while designing techniques to improve TCP performance in
3G/4G wireless networks have been extensively studied in the literature [54], our focus in
designing Q-MW is particularly on the impact of radio resource schedulers on TCP perfor-
ance. A closely related work is presented in [18], where the authors propose a TCP-aware
scheduling algorithm to reduce the latency of short TCP flows. Their solution requires a
detailed knowledge of TCP flows while our proposed solution requires only the knowledge of
user queues, which is readily available in LTE networks.

The other proposed scheduling algorithm is also based on MW and called H-MW. The
idea behind this scheduler is to study the effect of choosing the worst CQI to define the
modulation and coding scheme (MCS). Although different subcarriers allocated to a user
have various CQIs and are capable of transmitting data at different rates, due to hardware
limitations, MW chooses the worst CQI to define the associated MCS for the user. In other
words, suboptimal approaches are applied to assign resources. However, the suboptimal
approaches result in wasting radio resources and also decrease in the user’s total throughput.
To alleviate this decrease, H-MW is developed in which in each TTI, a subset of subcarriers
with the worst CQI is removed from allocated subcarriers to a given user if the achievable data rate of the associated user is improved in the current TTI. This, in turn, leads to leaving some of the subcarriers unassigned which may seem not desirable, but it results in a higher throughput for the users on average. While not studied in this thesis, this approach may potentially reduce network energy consumption as well by not allocating any power to channels with poor quality.

We study the performance of H-MW for different number of users using ns-3 with respect to TCP. Our simulation results show that the proposed H-MW outperforms the traditional MW in terms of the achieved throughput by 7% on average for TCP traffic. It should be noted that, when the ratio of the number of users to the number of resources is lower, the average number of allocated resources to each user increases; therefore, it is more likely to improve the achievable data rate of a user by removing a subset of subcarriers from the allocated subcarriers to a user.

1.3.3 Handover Effect on the Performance of Scheduling Algorithms

In the third part of the thesis, we focus on multi cell LTE networks and investigate the impact of handover on the performance of scheduling algorithms in these networks. A defining feature of cellular networks is the ability of users to roam in the coverage area of the network without losing their end-to-end connectivity. Indeed, achieving seamless user mobility, even at a high speed, is one of the prominent goals of LTE design. As a user moves from the coverage area of one cell to that of another cell, the old cell ”hands over” the user to the new cell. While the handover procedure in LTE is designed to have a low latency, there is still some time during which the user is disconnected from the network. To alleviate the service disruption during this time, the source eNB temporarily forwards the incoming data and the data that is already in buffer for the user to the target eNB. However, the forwarding of the user data may cause problems of its own when TCP is involved, due to an increased delay for the forwarded data. There can also be a time interval, immediately after the handover,
when packets on both the direct path and the forwarding path arrive in parallel, albeit with different delays, at the target eNB. This then may give rise to the problem of out of order packets and unnecessary reduction in TCP throughput due to the ensuing duplicate ACKs and spurious time-outs.

In LTE networks, to increase the spatial reuse of the system, and consequently increase the wireless bandwidth, small cells in the form of micro, pico and femto cells are being deployed [59]. This reduction of cell size in LTE leads to an increase in handovers for mobile users. It is thus critical to understand how handovers affect the LTE system, not only at the TCP layer, but also at other layers of the protocol stack. The impact of handovers on TCP performance in cellular networks has been extensively studied in the literature [62]. And there have also been several recent works specifically on this subject in LTE networks (e.g., [48]). However, the impact of handovers on the performance of radio resource schedulers is largely unknown.

In this part of the thesis, our goal is to study the performance of PF and MW with and without handovers in LTE networks. We measure the performance in terms of the average TCP throughput achieved under each scheduler. While there is significant work on the performance of PF and MW in the absence of handovers, it is extremely hard to capture the effect of handover on the scheduling algorithms using analytical models. Thus, in this work, we use ns-3 and different fading models to simulate realistic LTE networks across a wide range of network conditions in terms of round-trip delay, buffer size and channel fading. Although our results in the first part of the thesis show that MW achieves a higher throughput than PF when users are confined to a single cell, based on the results of this part specifically when there is handover in the network across multiple cells, PF achieves a throughput similar to that of MW, and in some cases even slightly outperforms MW.
1.4 Thesis Objectives and Contributions

The main objectives of this thesis can be summarized as follow:

- Study the behaviour of existing schedulers with TCP in LTE networks. Three variations of MW are implemented, their performances are studied in a wide range of network conditions, and compared with other schedulers built-in ns-3, such as PF, Round Robin (RR) and Frequency-Domain Maximum Throughput (FDMT).

- Proposing new schedulers. Two schedulers are designed and implemented based on MW to improve TCP throughput namely Q-MW and H-MW. To the best of our knowledge, queue-based scheduling has been theoretically studied in the literature, but it has not been implemented in an OFDMA LTE network. The idea in Q-MW is that those users whose packet queues at the eNB node are below a certain threshold are given higher priority regardless of their channel and queue conditions. The performance of this scheduler is studied in a wide range of network scenarios, in terms of queue size at the base station and round-trip delay.

  H-MW is another proposed scheduling algorithm in which a subset of subcarriers allocated to a user which has the worst channel quality is left unassigned if the achievable data rate of the associated user is improved in the current TTI. The performance of this scheduler is compared with MW in terms of the average TCP throughput for different number of users.

- Study handover impact on the performance of scheduling algorithms in LTE networks. The performance of PF and MW with and without handovers is studied and measured in terms of the average TCP throughput achieved under
each scheduler and investigated across a wide range of network scenarios in terms of round-trip delay, buffer size, and channel fading.

1.5 Thesis Outline

The remainder of the thesis is organized as follows: Chapter 2 surveys well-known scheduling algorithms introduced for the downlink of LTE systems. Chapter 3 provides a brief introduction to LTE with a specific focus on downlink scheduling and provides the background information necessary for understanding the concepts discussed in other chapters. Chapter 4 presents a brief introduction of the LTE simulation tools. Then, the ns-3 network simulator and LENA, the LTE module of ns-3, are explained and their functionalities and structures are discussed. Besides, this chapter provides descriptions on how to implement a simple LTE network. In Chapter 5, we provide an explanation on how to change an existing and/or implement a new scheduling algorithm in ns-3. Moreover, three implemented variations of the Max Weight scheduling policy are presented. The proposed scheduling algorithms, namely Q-MW and H-MW are included in this chapter as well. The simulation results of the thesis and the analysis performed are organized in Chapter 6. The network topology and system parameters used in the thesis are also discussed in this chapter. Finally, Chapter 7 concludes the thesis and suggests several areas for future work.
Chapter 2

Related Work

In this chapter, we review different scheduling strategies introduced for the downlink of LTE systems. These algorithms are well-known schemes that are widely used in the literature and presented to ease understanding for the main approach in this thesis, and to provide insight into the design of TCP-aware schedulers. In general, scheduling algorithms can be classified into two groups: channel-unaware scheduling and channel-aware scheduling approaches. Due to the unstable nature of a wireless channel, channel-unaware scheduling approaches will never perform well in LTE networks. On the other hand, well-designed channel-aware scheduling strategies can achieve a high performance by allocating resources according to channel conditions. Channel-aware schedulers can further be subdivided on the basis of QOS support into QOS-unaware or QOS-aware channel-aware scheduling strategies. Moreover, two other groups of solutions, semi-persistent and energy-aware solutions, are also presented [16], which are not covered in this chapter as they do not directly relate to our work.

2.1 Channel-Unaware Scheduling Strategies

This set of strategies was originally introduced in wired networks where transmission media is error-free and time-invariant. Since they do not account for channel quality variations, they are referred to as channel-unaware. Although this unawareness make them inappropriate for wireless networks, they can be combined with, or used as, the basic idea for channel-aware schemes [16].
2.1.1 Round Robin Scheduler

In Round Robin (RR) algorithm, time resources are assigned to each user in equal portions and in circular order, to handle all users without priority. Although the goal of RR is the fair sharing of time resources, it is completely unfair in terms of user throughput in wireless networks, since it does not take into account experienced channel conditions. It is also inefficient as it ignores users’ achievable data rates.

2.1.2 Blind Equal Throughput Scheduler

Throughput fairness among users that are associated with the same base station, can be achieved with Blind Equal Throughput (BET) algorithm by using users’ past average throughput as a metric [29]. Let $\hat{i}_k(t)$ denote the UE that is scheduled on RBG $k$ at TTI $t$. The scheduling decision for BET is made as:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} \left( \frac{1}{T_j(t)} \right),$$

where, $T_j(t)$ is the past average throughput of UE $j$ until TTI $t$. To calculate $T_j(t)$, a moving average is used as expressed below:

$$T_j(t) = \beta \cdot T_j(t-1) + (1 - \beta) \cdot R_j(k,t),$$

where, $\beta (0 \leq \beta \leq 1)$ is the weight factor for the moving average and $R_j(k,t)$ denotes the maximum rate achievable by UE $j$ on RBG $k$ at TTI $t$.

As can be seen, BET allocates resources to users that have been previously served with lower average throughput. Therefore, users with a poor channel condition are allocated more resources than others, which in turn improves fairness, but results in reduced total system throughput.
2.2 Channel-Aware/QoS-Unaware Scheduling Strategies

This set of schedulers takes advantage of CQI feedbacks collected from users by the AMC module, which is located in the base station, to estimate wireless channel quality. There are two types of CQI feedback, namely wide-band CQI, which estimates the channel quality over all the bandwidth, and sub-band CQI, which is the channel quality value over a specific RB.

2.2.1 Maximum Throughput Scheduler

The Maximum Throughput (MT) scheduling algorithm aims at maximizing the overall throughput by assigning each RB to the user with maximum expected data rate in the current TTI. The scheduling decisions are made as follows:

\[ \hat{i}_k(t) = \arg \max_{1 \leq j \leq N} R_j(k, t), \]

This strategy may lead to starvation of users with poor channel conditions (e.g., cell-edge users), and performs unfair resource sharing. This scheduler is studied for LTE systems in [33].

2.2.2 Proportional Fair Scheduler

The Proportional Fair (PF) scheduling algorithm tries to find a balance between fairness and spectral efficiency by merging MT and BET algorithms. The scheduling decisions are made as follows:

\[ \hat{i}_k(t) = \arg \max_{1 \leq j \leq N} (R_j(k, t)/T_j(t)), \]

This balance is provided by using past average throughput as a weighting factor of the expected data rate, which results in serving the users in bad channel conditions within a certain amount of time. The idea of PF scheduling algorithm has been extended in many
works. In [34], PF multiuser scheduling for LTE systems has been formulated as an optimization problem while it is assumed the user average SINRs are fairly uniform. Due to high computational complexity of proposed optimization problem, which makes it impractical in practice, a suboptimal PF scheduler is also proposed at the cost of throughput degradation.

The Generalized Proportional Fair (GPF) strategy proposed in [68], applies two new parameters to modify the impact of the achievable data rate and past average throughput on the scheduling policy:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} \left( ((R_j(k, t))^\xi / (T_j(t)))^\psi \right)$$

The basic PF metric is a particular case of the GPF where $\xi = \psi = 1$. By setting $\xi = 0$, the GPF metric would become equal to the BET metric, meaning that fairness can be achieved by the system regardless of the channel conditions. On the other hand, setting $\psi = 0$ would bring GPF to a MT policy without fairness. Similar approaches are proposed in [52] and [37].

2.2.3 Throughput to Average Scheduler

Throughput to Average (TTA) scheduler can be considered as a combination of MT and PF [29]. Its metric is expressed as:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} (R_j(k, t)/R_j(t)),$$

where $R_j(k, t)$ is the achievable data rate for RB $k$ determined based on the sub-band CQI of that specific RB, while $R_j(t)$ is the achievable data rate determined based on the wide-band CQI.

The above metric performs by averaging resources evenly between the users while guaranteeing that the best RBs are allocated to each user. Here, the achievable throughput in the current TTI is used as a normalization factor of the achievable throughput on the consid-
ered $k$-th RB. It is evident from its metric that the higher the overall expected throughput of a user, the lower its metric on a single RB will be.

2.2.4 Joint Time and Frequency Domain Scheduler

To reduce the computational complexity of LTE scheduling algorithms, [51] presented a two step strategy in which two schedulers, one in the time domain (TD) and the other one in the frequency domain (FD), are used in series. In the first step, the TD scheduler selects a subset of active users. These users are then frequency multiplexed by the FD scheduler in the second step. Therefore the FD scheduler only has to consider frequency multiplexing of users chosen in the first step per TTI.

An interesting feature of this strategy is that a different scheduling algorithm can be applied at each step. In [14], the authors propose a two step scheduling algorithm where the PF algorithm is used in both time and frequency domains to obtain fair sharing of time resources among users and a good trade-off between spectral efficiency and fairness.

2.2.5 Delay Sensitive Scheduler

In this kind of scheduling, the average data delivery delay is considered as the main performance metric. In [22], a cross-layer algorithm is proposed as an optimization problem to minimize the overall average packet delay. The proposed algorithm is shown to provide a lower delay than other algorithms. A similar metric has been used in [39] and [60] in which utility-based cross-layer wireless resource management architectures are presented.

2.2.6 Buffer-Aware Scheduler

The main idea behind this scheduler is decreasing the buffer overflow probability using buffer status information reported by the user to eNB. The proposed algorithm in [26], called cross-layer buffer-aware and traffic-dependent (BATD) packet scheduling, aims at decreasing the packet dropping rate due to buffer overflow by consideration of queue (buffer) status at
MAC layer, while keeping the total system throughput at a high rate and guaranteeing certain fairness among users. Unlike BATD in which only non real time traffic has been considered, [38] uses a very similar joint PHY-MAC layer design where priority between real time service and non-real time service is also distinguished as a feature of the algorithm.

2.2.7 TCP-Aware Scheduler

Designing packet schedulers becomes more challenging for networks with TCP traffic sources. Due to the many special features of TCP, the performance of scheduling algorithms under TCP traffic source, can be very different compared to that of under UDP traffic; in fact TCP has been shown to perform poorly in unreliable wireless networks. Particularly, TCP often considers random losses of wireless error-prone links as congestion signals and reacts to them by reducing its sending rate (e.g., transmission rate). This effect is especially severe in cases where the end-to-end Round Trip Time (RTT) is high [42]. In addition, other wireless features such as bandwidth fluctuations and a high delay variability negatively affect TCP [63]. This is why designing TCP-aware algorithms in wireless networks becomes an important area of research, however not many works tackle this problem in LTE networks. Here, we introduce a few works which try to address the TCP-aware resource-sharing problem in OFDMA-based wireless networks.

TCP-aware scheduling was first introduced and studied over a CDMA network in [24]. Based on another work, authors in [41] proposed a dynamic OFDMA-based scheduling algorithm aiming to constrain the achievable rate on each flow proportionally to the TCP throughput of the corresponding flow. In fact, the main idea considers the end-to-end performance of TCP flows to allocate resources more optimally and more fairly to active TCP flows. To this end, the algorithm is presented as an optimization problem that aims to maximize the total throughput that is constrained to the weighted proportional rate with respect to the TCP theoretical throughput. Based on the results, the authors show the effectiveness of the proposed approach to increase fairness among TCP flows.
Another TCP-aware scheduling algorithm is presented in [42], which aims to balance the sum rate and the theoretical TCP throughput of active TCP flows. In this case, the optimization problem is specified with the objective of maximizing the sum rate while, at the same time, minimizing the difference between the allocated data rate and the theoretical TCP throughput.

Since a high proportion of TCP traffic is short-lived flows, in [40], the authors attempt to improve the performance of short-lived TCP flows. The idea is to protect short-lived TCP flows in the slow start phase which provides an opportunity for them to increase their congestion window and hence their sending rate. To do so, a state diagram with a different level of protection and data rate for various flows, based on their congestion window, is defined. The proposed scheduling algorithm adapts its target Bit Error Rate (BER) and the data rate of each flow in response to the state of the corresponding flow. To meet the goal of the algorithm, the scheduler sets a lower target BER for a TCP flow with a small congestion window.

This small number of proposed TCP-aware scheduling algorithms are mostly theoretical works, and they have no comprehensive simulation by which to compare schedulers. In this project, we focus on TCP-aware scheduling in OFDMA LTE networks and conduct extensive simulations to do a performance study in LTE networks.

2.3 Channel-Aware/QoS-Aware Scheduling Strategies

This kind of scheduling associates a set of QoS parameters for each flow to guarantee the data rate and/or the bounded delivery delay.

2.3.1 Token Bank Fair Queue Scheduler

The token Bank Fair Queue (TBFQ) [69] aims to guarantee the minimum rate of a connection and is inspired by the leaky-bucket mechanism [71]. In TBFQ, as it is implemented in ns-3,
the traffic flow of UE $i$ is specified by 4 parameters (Fig. 2.1):

- $t_i$: packet arrival rate (byte/sec)
- $r_i$: token generation rate (byte/sec)
- $p_i$: token pool size (byte)
- $E_i$: the counter that records the number of tokens exchanged between the token bank and the flow $i$

To balance the traffic of flows, a shared token bank ($B$) is used by TBFQ in which every token represents a data byte. If the token generation rate $r_i$ is less than the packet arrival rate $t_i$, flow $i$ has to borrow tokens from token bank according to a priority metric $E_i/r_i$, in which case $E_i$ is decreased by the number of borrowed tokens. Otherwise, if $r_i > t_i$, the extra tokens are stored in the token bank and $E_i$ is increased by the number of added tokens.
TBFQ also monitors the traffic by changing the token generation rate. It also controls for a few parameters per flow, such as the debt limit to prevent an UE from borrowing too many tokens from the token bank, the maximum number of tokens each UE can borrow, and the minimum number of tokens each UE must store in the token bank in order to borrow more tokens after reaching its debt limit.

2.3.2 Priority Set Scheduler

The priority Set Scheduler (PSS) \[43\] is a QoS-aware scheduler which works in both time and frequency domains and can guarantee the required data rate \[71\]. To control fairness among UEs, PSS uses a parameter called the Target Bit Rate (TBR) to distinguish low and high priority flows. Therefore, two sets of UEs based on the TBR are formed by TDPS (Time Domain Packet Scheduler):

- High priority set: containing UEs whose past average throughput is smaller than TBR; TDPS calculates their priority metric using the BET algorithm.

- Low priority set: containing UEs whose past average throughput is larger than (or equal to) TBR; TDPS calculates their priority metric using the PF algorithm.

Once a number of UEs with the highest metric in the two sets has been selected by TDPS, the FDPS (Frequency Domain Packet Scheduler) allocates a RBG to the UE with the largest metric by using one of the following metrics considered in the LTE package of \textit{ns-3}:

- Proportional fair scheduled (PFsch)

\[
\hat{M}_{sch_k}(t) = \max_{1 \leq j \leq N} \left(\frac{R_j(k, t)}{T_{sch_j}(t)}\right),
\]
• Carrier over interference to average (CoIta).

\[
\hat{M}_{coi_k}(t) = \max_{1 \leq j \leq N} (CoI(j, k)/ \sum_{k=0}^{N_{RBG}} CoI(j, k)),
\]

where \(T_{sch_j}(t)\) is similar to the previously defined past average throughput of UE \(j\), with the difference being that it is updated only when the \(j\)-th UE is actually served. \(CoI(j, k)\) is an estimation of the SINR on the RBG \(k\) of UE \(j\). Both PFsch and CoIta metrics are proposed in a way to decouple the TD and FD schedulers.
Chapter 3

LTE Primer

In this chapter, after a brief introduction, the general network architecture and interfaces of LTE are presented. Next, the new air interface is explained for FDD systems. This is followed by a description of how user data is scheduled on the air interface as it is a major task of the LTE base station. The chapter concludes with an overall description of the Radio Resource Management (RRM) procedures in LTE.

3.1 Introduction

The enormous increase in mobile data usage and the arrival of multimedia applications have motivated the 3rd Generation Partnership Project (3GPP) to work on the Long-Term Evolution (LTE) for cellular networks. LTE is a fourth generation high-speed wireless network that evolved from the Universal Mobile Telecommunication System (UMTS), which in turn evolved from the Global System for Mobile Communications (GSM). The main goals of LTE are spectral efficiency, a high data rate for different services (such as VOIP, streaming multimedia and video conferencing), flexible carrier bandwidth, and QoS support. The key performance criteria for LTE are shown in Table 3.1.

In November 2004, the 3GPP began work on 4G technologies, particularly on a work item called system architecture evolution (SAE) which together with LTE is responsible for the evolution of packet core network (EPC) which will support high bandwidth services at a high data rate. With this work, 3GPP wanted to create a global standard for 4G technologies for two reasons. First, to give an operator full freedom to choose a vendor and second to help in removing the separation between various players such as network operators and vendors involved in providing services to end users.
Table 3.1: LTE performance requirements\cite{27}

<table>
<thead>
<tr>
<th>Metric</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peak data rate</td>
<td>DL: 100 Mbps&lt;br&gt;UL: 50 Mbps&lt;br&gt;(for 20 MHz spectrum)</td>
</tr>
<tr>
<td>Mobility support</td>
<td>- Optimized for low mobility up to 15 kmph&lt;br&gt;- High performance for speed up to 120 kmph&lt;br&gt;- Maintaining connection up to 350 kmph</td>
</tr>
<tr>
<td>Control plane latency</td>
<td>&lt; 100 ms (for idle to active)</td>
</tr>
<tr>
<td>(Transition time to active state)</td>
<td></td>
</tr>
<tr>
<td>Control plane capacity</td>
<td>&gt; 200 users per cell&lt;br&gt;(for 5 MHz spectrum)</td>
</tr>
<tr>
<td>Coverage (Cell sizes)</td>
<td>5 -100 km with slight degradation after 30 km</td>
</tr>
<tr>
<td>Spectrum flexibility</td>
<td>1.25, 2.5, 5, 10, 15, and 20 MHz</td>
</tr>
</tbody>
</table>

The network architecture of LTE is depicted in Fig. 3.1. This network has three main components\cite{65}:

- User Equipment (UE). \textit{e.g.}, a smartphone or tablet which communicates to the LTE network over an OFDMA radio interface.

- Evolved UMTS Terrestrial Radio Access Network (E-UTRAN). The E-UTRAN handles radio communications between the mobile and the evolved packet core and has one component, the evolved base station, called eNodeB or eNB. Each eNB is a base station that controls the mobiles in one or more cells and is capable of fast resource allocation over time slots of 0.5 milli-seconds. The base station that is communicating with a mobile is known as its serving eNB.

- Evolved Packet Core (EPC). The EPC acts as the intermediate gateway between the radio network (eNB) and the Internet and performs functions such as QoS control, charging, anchor point etc.
3.2 LTE System Architecture

The LTE network architecture, in general, is similar to that of GSM and UMTS. On the whole, the network is divided into two parts: a radio network and a core network. In contrast to older systems, the number of logical network nodes, has been decreased in order to streamline the overall architecture and to decrease cost and latency in the network [55]. Fig. 3.1 gives a graphical overview of the LTE network and its components.

In the LTE architecture, E-UTRAN has an important role as the air interface and is accompanied by SAE, which contains EPC network [17]. LTE and SAE together form the Evolved Packet System (EPS). The LTE network also uses an eNB, a SGW (serving gateway), a PGW (packet data network gateway), a MME (mobile management entity), and a HSS (home subscriber server) all of which are considered as part of the EPC except eNB (As shown in Fig. 3.1). The following subsections give an overview of the tasks and interactions of the different parts. We suggest the reader to refer to [56] and [19] for more details on the entities and physical interfaces.

3.2.1 Serving Gateway

The principal function of the serving gateway (SGW) is to manage the user IP tunnels between eNB and packet data network gateway. The SGW is connected to eNB through S1-UP (S1- user interface) and to the PDN gateway through the S5-UP interface. S1 and S5 tunnels for an individual user are typically independent of each other but it can be modified
as required. The SGW is connected to MME through the S11 interface, which functions to create and modify the tunnels. The S11 interface uses GTP-C (GPRS tunnelling protocol-control) to transfer the messages sent by MME to SGW. By default, MME and SGW are defined independently but these entities can be defined on the same network node depending on the operator’s choice. To have them independent allows the wireless standardization bodies to work on the signalling traffic and user traffic independently. This was done because the additional signalling increases the load of the processors working on the signalling traffic, and on the other hand the rising user traffic demands the evolution of more network interfaces and routing capacity [58].

3.2.2 Packet Data Network Gateway

The functions of Packet data network gateway (PDN-GW) are as follows [58]:

- This is the gateway to the Internet and connects to the SGW through S5-UP interface and to the Internet through SGi interface. Through a forward direction, it takes user data packets from the SGW and sends them to the Internet through SGi interface. Through a backward direction, data packets are encapsulated into S5 GTP tunnel and moved towards the SGW which serves the intended user.

- The PDN gateway assigns IP addresses to the mobile devices when a subscriber switches ON his/her mobile device. In this process, the mobile device first sends its request to eNB which uses the S1-CP to forward it to MME. After authentication, MME requests the IP address from the PDN gateway via a control plane protocol. If the PDN gateway approves the request, it returns an assigned IP address to MME. Then MME sends the request to eNB which then sends it to the subscriber.

- The PDN gateway has a key place in international roaming scenarios. A
roaming interface is used to connect the GSM/GPRS, UMTS/HSPA, or LTE networks of different network operators from varied countries. For example, if a subscriber travels to a foreign country and wants to access the Internet then the foreign network the visitor is accessing queries the home network’s database to authenticate the request. Once the query is authenticated, a bearer is established and GTP user tunnel is created between the SGW of the foreign network the visitor is accessing and his/her home network PDN-GW over an S8 interface. This process in called home routing and has an alternative referred to local breakout [55].

3.2.3 Mobile Management Entity

Mobility Management Entity (MME) is of great importance in LTE EPC architecture and has many functionalities. MME is the main signaling node in the EPC and to meet increasing signaling load in the network, multiple MMEs can be grouped together. LTE MME is responsible for initiating paging and authentication of a mobile device. MME retains location information at the tracking area level for each user and then selects the appropriate gateway during the initial registration process. MME connects to the eNB and SGW through the S1-MME and the S11 interfaces, respectively. MME also plays a vital part in handover signaling between LTE and 2G/3G networks and a role in general interworking with other radio networks [2].

Since the LTE standard only supports packet switching with its all-IP network, voice calls and text messaging (which are typically handled by circuit-switched networks such as GSM and CDMA) cannot thus simply be mapped to LTE, and MME is designed in a way to support these services. The reason why LTE is designed only for packet switching is that it aims to provide seamless Internet Protocol (IP) connectivity between UE and the packet data network (PDN) without any disruption to the end users’ applications during mobility [17].
3.2.4 Home Subscriber Server/Policy Control and Charging Rules Function

Home subscriber server (HSS) is a database that stores the information of each and every user in the network. It also does the authentication and authorization of the users and services provided to them [58]. On the other hand, Policy Control and Charging Rules Function (PCRF) is responsible for policy control decision-making, and controls the flow-based charging functionalities in the Policy Control Enforcement Function (PCEF), which resides in the PGW [17].

3.3 LTE Air Interface and Radio Network

LTE air interface is a completely revised air interface compared to previous 3GPP wireless systems which has been designed specifically to overcome the effects of multipath fading. Despite the common practice of spreading one signal over the complete carrier bandwidth (e.g. 5 MHz) in previous 3GPP wireless systems, LTE instead uses Orthogonal Frequency Division Multiplexing (OFDM) that transmits the data over many narrowband carriers of 180 kHz each. Therefore, a data stream is split into many simultaneous slower data streams instead of a single fast transmission. As a consequence of using longer transmission steps, comparing to UMTS, a similar data rate in the same bandwidth is achieved while the multipath effect is greatly reduced [55].

In the following subsections, we take a quick look at the user and control protocol stack in LTE. This is followed by a description of how LTE enables the use of much larger bandwidths in the downlink direction. As the aim of this thesis is improving the performance of downlink scheduling, we focus primarily on downlink direction in this section.

3.3.1 LTE Radio Protocol Architecture

The radio protocol architecture for LTE can be divided into control plane architecture and user plane architecture as shown in Fig. 3.2. The user plane protocol stack between the eNB
and UE consists of the following sub-layers: PDCP (Packet Data Convergence Protocol), RLC (radio Link Control), Medium Access Control (MAC), and Physical Layer. The control plane includes the Radio Resource Control layer (RRC) which configures the lower layers of the protocol stack.

On the user plane, data packets generated in the application layer are passed down to be processed by protocols such as TCP, UDP and IP. While in the control plane, RRC protocol creates the signaling messages that are transferred between the eNB and UE. In both cases, the information is processed by PDCP, RLC protocol and MAC protocol, and is then passed to the physical layer for transmission.

By definition, packets received by a layer are referred to as Service Data Unit (SDU) while the packet output of a layer is named a Protocol Data Unit (PDU). For example, on the transmit direction of the user plane, the PDCP sends a PDCP PDU to the RLC which is called an RLC SDU. In reverse, on the receive direction, a layer sends a SDU to a higher layer which receives it as a PDU. In Fig. 3.3 different protocol layers and their location in the LTE architecture are illustrated.

On the user plane side, packets in the core network (EPC) are encapsulated in a particular EPC protocol and tunneled between the PGW and the eNB. Depending on the interface,
different tunneling protocols are applied, such as the GPRS Tunneling Protocol (GTP) which is used on the S1 interface between the eNB and SGW, and on the S5/S8 interface between the SGW and PGW. [65]

3.3.2 LTE Physical Layer for Downlink Transmission: OFDMA

A key feature of the LTE network is that the radio interface uses OFDMA for the LTE downlink (DL) to significantly enhance speeds above that of UMTS or EV-DO. OFDMA has features such as high robustness against frequency selective fading and high spectral efficiency, and also allows flexible bandwidth sharing, which we further elaborate on in this thesis.

In LTE, instead of transmitting a data stream at a very high speed over a single carrier
Table 3.2: Defined bandwidths for LTE \[55\]

<table>
<thead>
<tr>
<th>Bandwidth (MHz)</th>
<th>Number of subcarriers</th>
<th>FFT size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25</td>
<td>76</td>
<td>128</td>
</tr>
<tr>
<td>2.5</td>
<td>150</td>
<td>256</td>
</tr>
<tr>
<td>5</td>
<td>300</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>600</td>
<td>1024</td>
</tr>
<tr>
<td>15</td>
<td>900</td>
<td>1536</td>
</tr>
<tr>
<td>20</td>
<td>1200</td>
<td>2048</td>
</tr>
</tbody>
</table>

as in UMTS, OFDMA divides the data stream into many slower data streams that are sent over many carriers simultaneously. The benefit of this strategy is that transmission steps can be long enough to keep away from multipath transmission problems on fast data streams. Table 3.2 represents the number of subcarriers (i.e. data streams) used where the center carrier, which is always left empty, is not included. The number of subcarriers is directly proportional to the bandwidth available for the overall LTE carrier. For example, a total of 600 subcarriers are used with an overall signal bandwidth of 10 MHz. To further explain, the total data rate can be up to 600 times the data rate of each subcarrier.

An important feature of the LTE physical layer is orthogonality which means the side lobes of each subcarrier wave are exactly zero at the center of the neighbouring subcarrier. This property results in considerable bandwidth saving. To decode data sent or received in this way, a mathematical function, referred to as Inverse Fast Fourier Transformation (IFFT), is used. Basically, the input to an IFFT is a frequency domain signal that is transformed into a time domain signal. Since each subcarrier uses a different frequency, the receiver uses an FFT to illustrate which signal was sent on each of the subcarriers at each time instance \[55\].

LTE uses the following physical specifications for the subcarriers:

- subcarrier spacing: 15 kHz;

- length of each transmission step (OFDM symbol duration): 66.667 microseconds;
• standard cyclic prefix: 4.7 microseconds. The cyclic prefix is transmitted before each OFDM symbol to avoid intersymbol interference because of different lengths of multiple transmission paths. A longer cyclic prefix of 16.67 microseconds has been specified for those environments with highly diverse transmission paths. However, the drawback of using a longer cyclic prefix is a decreased user data speed because the symbol duration remains the same; consequently, fewer symbols can be transmitted at each time interval.

In summary, the physical layer transfers all data from the MAC transport channels over the air interface, and it takes care of: the link adaptation (AMC), power control, cell search (for initial synchronization and handover purposes) and other measurements (inside the LTE system and among systems) for the RRC layer [10].

3.3.3 LTE Physical Layer for Uplink Transmission: SC-FDMA

The drawback of OFDMA is its high Peak to Average Power Ratio (PAPR) in combining the signals from multiple subcarriers which makes it non ideal for uplink data transmission. Practically, in a radio transmitter circuit, the amplifier is required to support the peak power output needed to transmit the data; this value determines the power consumption of the device regardless of the current required transmission power level.

In OFDM, PAPR is very high since the maximum power is rarely used and the average output power needed for the signal to arrive at the base station is much lower. However, the total throughput of the device corresponds to the average power not the peak power. Hence, a low PAPR is desired to achieve a balance between the transmitter power requirements, and the achievable data rates [55].

Although a high PAPR is acceptable for a base station as power is abundant, for a battery-driven mobile device, the transmitter should be as efficient as possible. Therefore, 3GPP has decided to apply another transmission scheme, referred to as Single-Carrier Fre-
Figure 3.4: LTE resource blocks in time and frequency domains

quency Division Multiple Access (SC-FDMA). Compared to OFDMA, SC-FDMA contains additional processing steps, while on the transmitting and receiving sides the same physical parameters are used as for the downlink direction. Further details related to the uplink can be found in [70].

3.3.4 LTE Resource Allocation

In LTE, radio resources are allocated in both time and frequency domains (see Fig. 3.4). The smallest radio resource unit that can be assigned to a UE is called a physical Resource Block (RB). The role of the resource allocation module is to dynamically allocate available time-frequency RBs to different UEs efficiently for good system performance. OFDMA assigns subsets of RBs (not necessarily adjacent ones) to individual users. In the frequency domain, the system bandwidth is divided into multiple sub-channels of 180 kHz (i.e., groups of 12 OFDM sub-carriers) where each RB spans over one sub-channel [25].

The length of the smallest transmission unit on each subcarrier is 66.667 microseconds which also referred to as a symbol and several bits can be exchanged per symbol depending on the modulation scheme. When channel conditions are excellent, 64-QAM is used to transfer 6 bits ($2^6 = 64$) per symbol. Under conditions not so excellent, other modulation schemes, specifically, 16-QAM or QPSK (Quadrature Phase Shift Keying) are used to transfer 4 or 2 bits per symbol [55].

An RB in the time domain is half a Transmission Time Interval (TTI) which lasts for two time slots each of 0.5 ms in length. Each time slot corresponds to 7 OFDM symbols. The
time is divided into frames, and each frame consists of 10 consecutive TTIs (or subframes). In each TTI, UE measures the pilot signal from the serving eNB and periodically reports the CQI to the eNB. The scheduler then uses this indicator together with queue information to calculate how many RBs each user is allocated.

3.3.5 The LTE Channel Model in Downlink Direction

As in UMTS, all higher layer signaling and user data traffic are classified into channels, specifically logical channels, transport channels and physical channels as shown in Fig. 3.5. The goal is to provide different pipes for different kinds of data at the logical layer and to decouple the logical data flows from the physical channel properties [55].

At the logical layer, an individual logical Dedicated Traffic Channel (DTCH) is used to transmit each user’s data. But, on the air interface all dedicated channels are mapped to a single shared channel that spans over all resource blocks. This process has two steps; mapping the logical DTCHs of all users to a transport layer Downlink Shared Channel (DL-SCH) is the first step. Second, DL-SCH data stream is then mapped to the Physical Downlink Shared Channel (PDSCH). In each resource block, some symbols are allocated for purposes besides transmitting user data. In every subframe, once per millisecond, the scheduler module located in the eNB decides which RBs are allocated to which user.

In addition to multiplexing data streams from several users, the transport channels multiplex several logical channels of an individual user before they are ultimately mapped to a physical channel. As an example a UE that has been allocated a DTCH also requires a
control channel for the management of the connection to send the messages required, for example, to control handover, to compute neighbor cell measurements and channel reconfigurations. Therefore the DTCH and DCCH are multiplexed on the DL-SCH before being mapped to physical RBs, i.e., to the PDSCH. Moreover, even most of the cell-specific information that is transferred on the logical broadcast control channel (BCCH) is multiplexed on the transport downlink shared channel as demonstrated in Fig. 3.5.

In LTE, the Paging Control Channel (PCCH) like all higher layer data flows is ultimately mapped to the physical shared channel. Specifically, the PCCH is first mapped to the transport layer Paging Channel (PCH), which in turn is mapped to the PDSCH. This channel is used to inform dormant mobile devices of new IP packets arriving from the network side.

General mapping of all higher layer information to the shared channel occurs except in the transmission of a small number of system parameters used to synchronize mobile devices to the cell. These parameters are sent on the Physical Broadcast Channel (PBCH), which spans over three symbols on 72 subcarriers (= 6 RBs) in the middle of a channel every fourth frame.

3.3.6 Downlink Management Channels

As described before, most channels are mapped to an individual PDSCH, which spans over all resource blocks in the downlink direction with the exception of those symbols in each resource block statically allocated for other purposes. Therefore, a mechanism is needed to determine for each mobile user when, where and the type of data scheduled for them on the shared channel and which resource blocks are assigned to them in the uplink direction. This mechanism is done by the Physical Downlink Control Channel (PDCCH) messages.

The downlink control information uses the first one to four symbols across the complete carrier bandwidth in each individual subframe. Physical Control Format Indicator Channel (PCFICH), which takes 16 symbols, is transmitted to broadcast the exact number of symbols that are used for PDCCH messages. This enables the system to flexibly react on variable
number of users that are to be scheduled in a subframe.

Downlink control data is classified into Control Channel Elements (CCEs). One or more CCEs contain a single signaling message that is either sent to one mobile device, or, in the case of broadcast information, to all devices in the cell. In order to decrease the processing time and power consumption of mobile devices for decoding the control information, the control region is divided into search spaces. Consequently, each mobile device decodes the CCEs allocated to it however not all CCEs in the search space.

Besides, some symbols are reserved to signal to the mobile device if the uplink data blocks are received correctly or not. This functionality is done via the Physical Hybrid Automatic Retransmission Request Indicator Channel (PHICH) and called the Hybrid Automatic Retransmission Request (HARQ).

Overall, the PDSCH is transported in all resource blocks over the whole system bandwidth. However, there are some reserved symbols in each resource block which will be used for other things such as the broadcast channel, the control channel, the physical control format indicator channel, the synchronization signals and the HARQ indicator channel. Depending on the resource block and its location in the resource grid, the number of symbols not available to the shared channel is variant. A mathematical formula is provided in 3GPP TS 36.211 [64] for each signal and channel, which helps the mobile device to determine the location of a specific information [55].

3.4 Downlink Scheduling

In LTE, both the uplink and the downlink data transmissions are controlled by the network which is comparable to the other technologies such as GSM and UMTS [55]. The scheduling in LTE is entirely controlled by the eNBs since the higher layer radio network control entities were deleted from the network design. This network-based scheduling has some benefits as noted:
• The network can react to users’ variable channel conditions and optimize the overall throughput.

• The network can guarantee the QoS for users.

• The network can deal with overload situations.

In the downlink direction, this is the responsibility of the eNB’s scheduler to forward the data received from the network, for all users it serves, over the air interface. Practically, an individual logical default bearer is usually allocated to a mobile device, over which the data is transmitted. However, to guarantee QoS for applications such as VoIP via the IMS (IP Multimedia Subsystem), allocating more than one logical bearer to a mobile device is also feasible. Therefore the VoIP data stream can use a dedicated bearer for which the required bandwidth and a low time variation between two packets (jitter) is guaranteed by the network. There are two types of scheduling; dynamic and semipersistent. In the following, we focus on dynamic scheduling which is the more common type compared to semipersistent scheduling.

Scheduling is simple if there is only one user and if the amount of data in the downlink buffer is less than data that can be transmitted over the air interface. When the eNB serves several users, or more precisely several bearers, and there is more data waiting in the transmission buffer than can be sent in a subframe, then the scheduler has to define which users and bearers are allocated an assignment grant for the next subframe and the capacity amount given to each, in the next subframe. The scheduler’s decision depends on various factors.

The scheduler has to guarantee the requested requirements, when a particular bandwidth, delay and jitter have been ensured for a bearer to a specific user. Hence, this bearer’s data has priority over the data being received from the network of other bearers for the same or a different user. However, in reality, most bearers have the same priority and these kind of QoS requirements are not commonly used on the radio interface.
In the case where bearers have the same priority, there are other factors influencing the scheduler’s decision, specifically the time to schedule a user and the number of resource blocks that are allocated to the user in each subframe. There are two extremes; first, equally considering all bearers with the same priority in which case some capacity on the air interface would be wasted. Taking this strategy, users that currently or permanently experience poor channel conditions, for instance, at a cell edge, would need to be allocated an unbalanced number of resource blocks due to the required low modulation and coding scheme. Second, mobile devices experiencing excellent channel condition are always preferred which results in very low throughputs for users experiencing poor radio conditions. Because of this, proportional fair schedulers try to balance the best use of the cell’s total capacity and each user’s data rate by considering the overall radio conditions and observing channel fluctuations for each user over time.

Scheduling downlink data for a user is processed as follows: The eNB’s scheduler determines the number of users which will be scheduled and also the number of resource blocks that are allocated to each user, in each subframe. In turn, this defines the required number of symbols on the time domain for each subframe. As demonstrated in Fig. 3.4 in each TTI, there are in total $2 \times 7 = 14$ symbols available on the time axis provided that a short cyclic prefix is used. For the control region, one to four symbols are used over the whole channel bandwidth, depending on the system configuration and the number of scheduled users. The number of symbols used for the control region can be fixed or variable according to demand.

PDCCH messages categorized into several message types, consist of different lengths and can be used for many different purposes. In the technical requirements, the message type is called Downlink Control Information (DCI) format.

The message describing a user’s downlink assignment on the downlink shared channel includes the following information:

- the type of resource allocation, type 0, 1 or 2;
• a power control command for the PUCCH message;

• HARQ related information such as new data bit, redundancy version;

• a modulation and coding scheme such as QPSK or QAM;

• a number of spatial layers used in MIMO;

• precoding information such as the data preparation steps before transmission.

The LTE downlink supports three resource allocation types:

• Type 0 resource allocation: several consecutive RBs make up a resource block group (RBG), and the smallest resource allocation unit is a RBG. This type gives a bitmap specifying the allocated RBGs to define the resource assignment. The RBGs allocated to a certain UE are not necessarily adjacent, which results in frequency diversity. The number of resource blocks in a RBG, named as the size of RBG P, depends on the bandwidth and is specified in Table 3.3 [25].

• Type 1 resource allocations: like type 0 uses a bitmap, however, from each RBG only one of the RBs is allocated to UE instead of allocating the whole RBG. This approach leads to spreading resource assignments across the whole bandwidth which provides more flexibility and higher frequency diversity, however, this means a larger overhead.

• Type 2 resource allocation: determines a beginning point in the frequency domain and the number of allocated resources which can either be continuous or span across the entire channel.
Table 3.3: Resource allocation RBG size vs. downlink system bandwidth

<table>
<thead>
<tr>
<th>Downlink Resource Blocks</th>
<th>RBG Size (P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>≤ 10</td>
<td>1</td>
</tr>
<tr>
<td>11 - 26</td>
<td>2</td>
</tr>
<tr>
<td>27 - 63</td>
<td>3</td>
</tr>
<tr>
<td>64 - 110</td>
<td>4</td>
</tr>
</tbody>
</table>

3.5 RRM Procedures in LTE

In addition to resource distribution, LTE takes advantage of RRM procedures such as link adaptation, HARQ, etc. These procedures that are located at physical and MAC layers, work in tandem to improve the usage of available radio resources. In this section, a brief description of two important RRM procedures are presented.

3.5.1 Adaptive Modulation and Coding (AMC)

LTE employs adaptive modulation and coding (AMC) to derive a benefit from channel fluctuations over time and frequency. The main idea of AMC procedure is: transmitting at the highest possible data rate when and where the channel is good, and transmitting at a lower rate when and where the channel is poor to prevent dropping more packets. More precisely, Adaptive Modulation and Coding is the LTE network’s ability to dynamically indicate the modulation type and the coding rate depending on the UE’s channel conditions reported to the eNB in Measurement Reports. The digital modulation and coding schemes used to transmit the information are QPSK, 16-QAM, and 64-QAM. To transport at a lower data rate, a small, low-order, constellation, such as QPSK, and low rate error correcting codes such as rate 1/3 turbo codes are applied. The higher data rates are attained by using large, high-order, constellations, such as 64QAM, and weaker error correcting codes, for example, either higher rate (like 3/4) codes, or punctured turbo codes.
3.5.2 Hybrid Automatic Repeat Request (HARQ)

With adaptive modulation and coding schemes, it is hard to ensure that all of the transmitted data packets are received correctly. In fact, it is even favorable that not all packets are received correctly since this would mean that the modulation and coding scheme is too conservative which in turn leads to wasting capacity on the air interface. In reality, the channel bandwidth is optimally utilized when around 10% of the packets are retransmitted due to being received in error [55].

To this end, ARQ (Automatic Repeat Request) and Hybrid-ARQ are the techniques that are often used in modern wireless communication systems including LTE. ARQ is referred to as a MAC layer retransmission protocol that allows packets received in error to be speedily retransmitted. This protocol works cooperatively with PHY layer ECCs (Error Correction Code) and parity checks in order to ensure reliable links even in unreliable channels. Since a single bit error causes a packet error, the ARQ mechanism has to resend the entire packet even if most of the bits already received were correct, which is undoubtedly inefficient. To illustrate, if the same packet is dropped twice in a row, while 99% of its bits were correctly received, it is quite possible that every bit was received correctly at least in one of the two packets.

To avoid such a waste, Hybrid-ARQ merges the two ARQ and FEC strategies, by combining received packets. With this approach, Hybrid-ARQ, is also able to exploit additional time diversity in a fading channel. In H-ARQ, the main idea is using a channel encoder such as a convolution encoder or turbo encoder to add extra redundancy to the information bits. However, just a fraction of encoded bits (original bits + redundancy bits) are transmitted, not all of the encoded bits. This is done by puncturing some of the encoded bits to generate an effective and greater code rate compared to original encoder code rate. Code rates are defined as fractions in which the number of data bits and total number of bits are specified in the numerator and denominator, respectively. So, for instance, if the Code Rate is $1/3$, 2
protection bits are added so three bits in total are sent for one bit of data. After receiving the encoded and punctured bits, the receiver sends an acknowledgment to the transmitter telling it if it could successfully decode the information bits. If the receiver was able to decode the information bits, no extra action is needed. Otherwise, the transmitter can resend another copy of the encoded bits to the receiver [25].

For this approach to be successful, the transmission errors need to be reported as quickly as possible and also the packets need to be retransmitted quickly to minimize the resulting delay and jitter. Moreover, the modulation and coding scheme must be speedily adapted by scheduler to keep the error rate within reasonable limits [55].
Chapter 4

LTE Simulation Tools

In this chapter, we first briefly introduce and compare some of the well-known network simulation tools. Then, ns-3 network simulator is explained and its functionality and modules are discussed in detail. Next, the LTE module of ns-3 is described. Specifically, we discuss the LTE functionality and structure. This is followed by a description of how to implement a simple LTE network in ns-3.

4.1 Simulation in Networking Area

Simulation is a useful widespread technology that can be applied in networking area to model different aspects of networks such as network traffic. Network simulation means using computer-assisted simulation technologies in the simulation of networking algorithms or systems [49].

Diverse groups such as academic researchers, industrial developers, and Quality Assurance (QA) system providers use network simulators to design and model network topologies to verify and analyze the performance of various network protocols. Network simulators can also be used to assess the effect of varying parameters on network protocols. Basically, a network simulator consists of a wide range of networking technologies and protocols and enables users to design complicated networks by using basic building blocks such as groups of nodes and links. In other words, using a network simulator, one can model real world networks in order to change different parameters and then analyze the corresponding results. Therefore, a wide variety of differing scenarios can be analyzed at a low cost to give the user meaningful insight into the network being simulated.

Currently, there are different commercial and open source network simulators. For exam-
ple commercial simulators such as OPNET [7] and QualNet [8] and open source simulators such as ns-2 [4], ns-3, OMNeT++ [6] and SimPy [9] are widely used. As choosing the appropriate network simulator is a vitally important task, several performance comparison studies are presented to evaluate network simulators for different network parameters and protocols [67] [31] [32] [53]. Based on these evaluations, it can be concluded that selecting a network simulator is dependent on a variety of parameters such as the scenario and metric one is interested in.

After considering the pros and cons of existing network simulators, we concluded that ns-2, the most popular simulator for research usage [32], was a smart choice for this project. However, after spending a significant amount of time working with ns-2, we discovered that this simulator lacks several essential functionalities. Specifically, considering the ultimate goal of this project, ns-2 lacks AMC module, while the existence of this module is vital for OFDMA-based networks. Therefore, we switched to ns-3 which had demonstrated a good overall performance compared to other simulators, based on the aforementioned research studies. It should be noted that ns-3 is not an extension of ns-2, however, because there were problems with the ns-2 design, the ns-3 is now under development and test [49]. Some models from ns-2 have already been ported to ns-3 and many ns-2 issues have been resolved in this simulator. Therefore, ns-3 is a new and different simulator that can be considered as a replacement for ns-2 [31].

Our main reason for choosing ns-3 was to have access to an almost comprehensive set of models and modules developed for this simulator. In addition, ns-3 performs well in terms of simulation run-time and scalability, and shows both low computational and less memory demands [67] [53].
4.2 *ns-3* Simulator

*ns-3* is a discrete-event network simulator for Internet systems, targeted principally for research and educational use that is written completely in C++ [15]. It is, an open-source project which started in 2006 and is licensed under the GNU GPLv2 license. *ns-3* provides models of how packet data networks work and perform, and provides a simulation engine for users to conduct simulation experiments. *ns-3* is mainly used on Linux systems and several external animators and data analysis and visualization tools can be used with it. *ns-3* is being actively developed and has a users’ mailing list for general discussions about the simulator.

Simulation *ns-3* programs can be coded in C++ or Python. Most of the *ns-3* API is available in Python, but the models are written in C++. The *ns-3* project uses Mercurial as its source code management system and the build system Waf to compile a source code to produce executable simulation programs.

Since *ns-3* is distributed as source code, the target system must have a software development environment to build the libraries and then develop the user program. We do not want to delve into the *ns-3* installation process, but the details on how to download, build and test *ns-3* can be found in the third chapter of [15].

Overall, the main features of *ns-3* can be summarized as follows [35][15]:

- C++-only simulations; easy debugging
- Scalable core, thereby minimizing the coupling between different models
- Real-world integration, simulating real system models
- Software integration, easier to interact with other software packages
- Support for virtualization and testbeds; can be used as a virtual system or in testbeds which makes *ns-3* more of an emulator
4.2.1 Modular ns-3

The ns-3 simulator provides several modules and models and it is important to know the difference between the two concepts [46]:

- Module: An important feature of ns-3 is that it is modular. Each module is built as a different software library and the required modules (libraries) can be linked in an individual ns-3 program to conduct a simulation. Based on the latest version of the simulator at the time of this writing, ns-3.19, the ns-3 is organized into over 30 modules, such as lte, network, internet, applications, mobility, etc. A list of all modules and their descriptions can be found in the ns-3 documentation [5] that is maintained using Doxygen.

- Model: Real-world objects, protocols, devices and more are modeled with abstract representations in ns-3.

An ns-3 module may contain more than one model (for example, the internet module includes models for both TCP and UDP protocols). However, ns-3 models do not commonly spread over multiple modules.

In [46], a documentation about the models of ns-3 is provided in which the models are categorized based on their associated module.

To have a better understanding of how different modules are organized in the ns-3 source code, Fig. 4.1 represents a schematic view of ns-3 software organization [44]. Most of the ns-3 source code is organized in the src directory and modules depicted in the figure are only dependent on the modules situated below them. The lowest module, the core module implemented in the src/core, includes those components in common among all protocol, hardware, and environmental models. The network module, whose source code lives in the src/network, contains the implementation of Packets that are basic objects in a network.

\footnote{Doxygen is typically used for API documentation, and classifies such documentation across different modules. [5]}
simulator. These two simulation modules, namely core and network, by themselves can make up a generic simulation core that can be used by various kinds of networks, not only Internet-based networks. The modules above the network module in figure Fig. 4.1 do not depend on any specific network and device models. The other fundamental ns-3 objects are: Node, Application, Socket, NetDevice (a network card which can be plugged in an IO interface of a Node) and Channel [35].

4.2.2 Helper/Container APIs

To build high-level topologies, the main approach of ns-3 is to define several helper classes that work in conjunction with containers [35]. The basic idea is quite simple: one operation is implemented in a Helper object and is applied on a Container which is a cluster of objects. This approach simplifies building topologies with duplicating patterns and also results in a more high-level topology description which in turn makes it easier to read and understand LTE code. In other words, in a ns-3 network simulation, it is very common to arrange and connect NetDevices to Nodes, NetDevices to Channels, assign IP addresses, etc. which may take many operations. Topology helper objects are provided to help combine these many distinct operations into an easy-to-use model [45].

It should be noted that Helper operations are not generic, i.e., different helpers provide different operations. Besides, they do not allow for code reuse and, in fact, merely try to
minimize the amount of code written. In \textit{ns-3} each model provides a helper class; some examples of helper classes are \textit{InternetStackHelper}, \textit{WifiHelper}, \textit{MobilityHelper}, etc. Moreover, \textit{NodeContainer}, \textit{NetDeviceContainer}, \textit{Ipv4AddressContainer} are some container examples.

4.2.3 Logging and Tracing in \textit{ns-3}

To generate a structured form of output, the \textit{ns-3} provides a tracing and statistics gathering system with no need to rebuild the simulation code \cite{49,35}. In fact, the tracing system decouples tracing as much as possible from the simulation code itself. In this system, tracing hooks predefined by the code are connected to tracing sinks the user makes. A tracing hook can be, for instance, the occurrence of a certain error state, the arrival of a packet, etc. Depending on the user’s needs, tracing sinks can analyze the incoming data, accumulate it and store it in differing formats \cite{15}.

A statistics module is also included in \textit{ns-3} which contains some useful features to ease data collection from experiments. The statistical framework of \textit{ns-3} is implemented in the directory \texttt{src/stats}. Some of the module features are: two basic data collectors (a counter, and a min/max/avg/total observer), extended data collectors for times and packets, plain-text output formatted for OMNet++, and database output provided using SQLite (a SQL engine) \cite{47}.

4.3 LTE Package of \textit{ns-3} Simulator

In this thesis, we use the LTE component of the \textit{ns-3} simulator called LENA\footnote{We use \textit{ns-3.18} and \textit{ns-3.19} which at the time of conducting simulations and writing are the latest releases.} to accurately simulate the dynamics of the LTE radio interface \cite{50}. The LENA module closely emulates 3GPP standards for the data plane, faithfully reproducing interactions of the EPC as well as the various stacks of the LTE radio layer such as the PDCP, RRC, MAC and PHY. It incorporates the Femtocell scheduling framework \cite{23} that reproduces how scheduling takes
place according to the 3G framework, as well as generating accurate models for emulating transmission, fading and decoding on the radio interface. As such, the simulator provides a fairly accurate proxy for the actual network. In the following, we describe the module in more detail.

As mentioned previously, LENA is a LTE/EPC simulation model which has two main components [20] as can be seen in Fig. 4.2:

- the LTE Model. This model contains the LTE Radio Protocol stack (RRC, PDCP, RLC, MAC, PHY). These entities live totally within the UE and the eNB nodes.

- the EPC Model. This model contains core network interfaces, protocols and entities. These entities and protocols live in the SGW, PGW and MME nodes, and partly within the eNB nodes.

The main design approach in LENA is to minimize implementation complexity to make the simulator easy to use. To this end, some design choices have been made for each of the LTE and EPC models. Following are some of the important design choices [20]:

- The LENA introduces support for the LTE MAC Scheduler API in the simulator to more accurately represent the LTE standard.
• Radio signal model granularity is one RB which simplifies the Channel and PHY models. However, it also makes system level simulation impossible.

• The LENA provides a real-world implementation of the EPC Data Plane Protocol stack model (Realistic RLC, PDCP, S1-U, X2-U) which, in turn, allows proper interaction with IP networking. On the other hand, the EPC control plane is simplified; In fact, realistic RRC model is implemented while S1-C, X2-C and S11 models are simplified.

• The SGW and PGW functional entities are implemented within a single node to simplify EPC (no S5/S8 interface).

In the following, we describe some of the LENA module features related to our work.

4.3.1 Channel and Propagation

Regarding channel modeling, the LTE module requires the use of the MultiModelSpectrumChannel to work properly [20] because the LTE must support different frequency and bandwidth configurations. The Buildings module, which was originally designed for LTE, provides a propagation model to use with the LTE. It should be noted that the LTE module considers FDD only, and implements downlink and uplink propagation separately.

Besides, the LTE module contains a trace-based fading model. In this model, the fading evaluation during simulation run-time is based on pre-calculated traces which are done to limit the computational complexity of the simulator. The mathematical channel propagation model used to generate these fading traces is the rayleigh model which provides a well-accepted channel characterization both in the time and frequency domains. The simulator also includes a Matlab script to generate traces based on a user’s desired parameters. Traces for 3 different scenarios, namely Pedestrian, Urban and Vehicular are provided in the simulator. Since the LTE PHY model supports antenna modeling via the ns-3 AntennaModel class, any model based on this class can be associated with any eNB or UE instance,
such as the CosineAntennaModel or the IsotropicAntennaModel (the default model used for both eNBs and UEs).

4.3.2 PHY Model

The physical layer model provided in the LTE simulator includes the Gaussian inter cell interference calculation and the uplink traffic simulation, including both packet transmission and CQI feedback. Two types of CQI feedback are provided in LENA: wide-band CQI which estimates the channel quality over all the bandwidth, and sub-band CQI, which represents the channel state for a specific RB. Other characteristics of the LENA PHY model are as follows:

- The time domain granularity is 1 TTI which is further divided into DL and UL, as can be seen in the LTE subframe division in Fig. 4.3.

- To model channel losses, the simulator includes an error model of the data plane (i.e., PDSCH and PUSCH), and also the error model for the downlink control channels (PCFIC and PDCCH), while in uplink it is assumed to be an ideal error-free channel. The error model is based on a link-to-system mapping (LSM) technique.

- The LENA PHY layer implements both the SISO (Single-input Single-output) and MIMO (multiple-input multiple-output) models where MIMO is modeled as SINR gain over SISO.

4.3.3 Hybrid ARQ

The implemented HARQ scheme is the IR Type II which is based on an incremental redundancy (IR) solution combined with multiple stop-and-wait processes for enabling a continuous data flow. Retransmissions are managed by the scheduler implemented within the
respective scheduler classes. According to the standard, the UL retransmissions are synchronous while the DL retransmissions are asynchronous. Finally, HARQ is integrated with the PHY error model, i.e., \texttt{LteMiErrorModel} class.

### 4.3.4 MAC Layer Model

#### Resource Allocation Model

In LENA, the packet scheduler is in charge of distributing the available resources among active UEs and is deployed at the eNB. The implementation of LTE schedulers in LENA follows the LTE MAC scheduler interface specification [23] as defined by Small Cell Forum [59]. The scheduler generates specific structures called the Data Control Indication (DCI) which are then transmitted by the PHY of the eNB to the connected UEs, to inform them of the resource allocation on a per sub-frame basis. To do so, the scheduler fills specific fields of the DCI structure with various control information, such as the Modulation and Coding Scheme (MCS) to be used, the MAC Transport Block (TB) size, and the allocation bitmap which identifies which RBs will contain the data transmitted by the eNB to each user.

For the mapping of resources to physical RBs, LENA applies a localized mapping approach; hence, in a given sub-frame, each RB is always allocated to the same user in both
slots. The allocation bitmap considered in LENA implementation is Allocation Type 0, according to which the RBs are grouped in Resource Block Groups (RBG) of different size which are defined as a function of the Transmission Bandwidth Configuration in use. In other words, the allocation type defines the RBG size (the number of RBs for a RBG) based on the system bandwidth (total number of downlink RBs) [20].

**Adaptive Modulation and Coding**

Two AMC algorithms working on reported CQI feedbacks are provided, namely Piro and Vienna. The Piro model is a conservative model based on an analytical BER. The Vienna model aims for a maximum of 10% Block Error Rate (BLER) based on the error model curves generated via system level simulations.

**MAC Scheduler Implementations**

The LENA scheduler implementations are all based on the FemtoForum API as follows: Round Robin (RR), Proportional Fair (PF), Maximum Throughput (MT), Throughput to Average (TTA), Blind Average Throughput, Token Bank Fair Queue, Priority Set Scheduler, Channel and QoS Aware Schedulers. A description of most of these schedulers is provided in Chapter 2. It should be noted that the above algorithms are for the downlink only; however for uplink, all current implementations use the same Round Robin algorithm.

### 4.4 Implementing a Simple LTE/EPC Network

In this section, we walk through a simple LTE/EPC example program. The source code for this example is included in Appendix A. This script instantiates one eNBs and attaches one UE to it. It also creates an EPC network and a remote host connected to it through the Internet. The script starts a flow for the UE to and from the remote host.

A ns-3 code begins by loading the modules’ include files. In fact, a single include file will recursively load all of the include files used in each module to enable a user to load a
group of files at a large granularity \cite{49}. Then, the ns-3 namespace is used and a logging component is defined as follows:

```cpp
using namespace ns3;
NS_LOG_COMPONENT_DEFINE ("LTE/EpcExample");
```

The main part of the script is included in a `main` function. At the beginning of the function, a number of local variables are initialized such as number of nodes and simulation time:

```cpp
int main (int argc, char *argv[]) {
    uint16_t numberOfNodes = 1;
    double simTime = 1.1;
    double interPacketInterval = 100;
}
```

Then, objects of `lteHelper` and `epcHelper` are created. The first line instantiates some common objects (e.g., the Channel object) and provide the methods to add eNB and UE and configure them. Applying EPC allows the use of IPv4 networking with LTE devices \cite{20}. Since the `EpcHelper` class is an abstract base class, we need to use one of its child classes. Here, `PointToPointEpcHelper`, which implements an EPC based on point-to-point links, is considered. Finally, we must let the LTE helper know that the EPC will be used as noted below:

```cpp
Ptr < LteHelper > lteHelper = CreateObject < LteHelper > ();
Ptr < PointToPointEpcHelper > epcHelper =
    CreateObject < PointToPointEpcHelper > ();
lteHelper->SetEpcHelper (epcHelper);
```

Although `EpcHelper` automatically creates the PGW node and configures it so that it can properly handle traffic from/to the LTE radio access network, some code must be added to connect the PGW to other IPv4 networks (e.g., the Internet). The following code outlines
how to connect a single remote host to the PGW via a point-to-point link. Note that to create an external host, we use NodeContainer topology Helper which simplifies the creation, management and access to any Node object.

```cpp
Ptr<Node> pgw = epcHelper->GetPgwNode ();

// Create a single RemoteHost
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);

InternetStackHelper internet;
internet.Install (remoteHostContainer);

// Create the Internet
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate",
    DataRateValue (DataRate("100 Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);

Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces =
    ipv4h.Assign (internetDevices);

// interface 0 is localhost, 1 is the p2p device
Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);
```

In the next step, we need to enable the remote host to reach the LTE UE by specifying routes. Considering the fact that the PointToPointEpcHelper will, by default, assign to
the LTE UEs an IP address in the 7.0.0.0 network, the following code will take care of this:

```cpp
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
    ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address("7.0.0.0"),
    Ipv4Mask("255.0.0.0"), 1);
```

Now, we should create LTE eNB and UE as explained before for the remote host and
Install the LTE protocol stacks on the nodes:

```cpp
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create(numberOfNodes);
ueNodes.Create(numberOfNodes);

// Install LTE Devices to the nodes
NetDeviceContainer enbLteDevs =
    lteHelper->InstallEnbDevice (enbNodes);
NetDeviceContainer ueLteDevs = lteHelper->InstallUeDevice (ueNodes);
```

The following code defines the node positions and configures their Mobility model:

```cpp
Ptr<ListPositionAllocator> positionAlloc =
    CreateObject<ListPositionAllocator> ();

positionAlloc->Add (Vector(0, 0, 0));

MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator(positionAlloc);
mobility.Install(enbNodes);
```
mobility.Install(ueNodes);

By using the following part of the code, we can configure the UE for IP networking as:

```cpp
internet.Install (ueNodes);
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address(NetDeviceContainer(ueLteDevs));
// Assign IP address to the UE, and install applications

Ptr<Node> ueNode = ueNodes.Get (0);
// Set the default gateway for the UE
Ptr<Ipv4StaticRouting> ueStaticRouting =
ipv4RoutingHelper.GetStaticRouting (ueNode->GetObject<Ipv4> ());
ueStaticRouting->SetDefaultRoute
  (epcHelper->GetUeDefaultGatewayAddress (), 1);
```

As explained, we want to attach the only UE to the eNB; this can be done as follows:

```cpp
lteHelper->Attach (ueLteDevs.Get (0), enbLteDevs.Get (0));
```

To install applications on the LTE UE node that communicate with remote applications over the Internet, the usual ns-3 procedures must be followed. Below is the code to setup downlink and uplink communication between the remote host and the UE with UdpClient applications on the remote host, and PacketSink on the LTE UE for downlink and vice versa for uplink:

```cpp
uint16_t dlPort = 1234;
uint16_t ulPort = 2000;
ApplicationContainer clientApps;
ApplicationContainer serverApps;
```
To setup TCP communication, TcpSocketFactory is used instead of UdpSocketFactory to create PacketSink. Also a number of different helpers can be used to create a client such as BulkSendHelper which creates a very common ns-3 TCP application namely Bulk TCP Transfer. This helper manages to send back-to-back TCP packets for the entire duration of
the simulation or a limited time:

```cpp
PacketSinkHelper dlPacketSinkHelper("ns3::TcpSocketFactory",
    InetSocketAddress(Ipv4Address::GetAny(), dlPort));
BulkSendHelper dlClient("ns3::TcpSocketFactory", InetSocketAddress
    (ueIpIface.GetAddress(0), dlPort));
```

Finally, the last steps are setting the stop time, starting the simulation and cleanup:

```cpp
Simulator::Stop(Seconds(simTime));
Simulator::Run();
Simulator::Destroy();
return 0;
```
Chapter 5

TCP-Friendly Schedulers

In this chapter, we first provide an explanation on how to change an existing and/or implement a new scheduler in ns-3. Then, three variations of the Max Weight scheduling policy that we implemented in ns-3 are described. This chapter concludes with the presentation of the proposed scheduling algorithms, namely Q-MW and H-MW.

5.1 Implementing a New Scheduler in ns-3

Before going through the details of a new scheduler’s implementation, following is a brief description of the specification of the LTE MAC Scheduler Interface implemented in ns-3. The FemtoForum MAC Scheduler Interface primitives are classified into two groups: the CSCHED primitives which handle scheduler configuration, and the SCHED primitives which manage the execution of the scheduler [20]. In another classification, the primitives are type REQ which travel from the MAC to the Scheduler, or are type IND/CNF which move from the scheduler to the MAC. To issue the primitives, 4 abstract classes are defined to implement Service Access Points (SAPs) in ns-3, specifically the FfMacSchedSapProvider, the FfMacSchedSapUser, the FfMacCschedSapProvider and the FfMacCschedSapUser. All the C++ methods that correspond to the primitives of type REQ and CNF/IND are defined in the provider and user classes respectively.

Fig. 5.1 demonstrates how the MAC Scheduler Interface is implemented within the eNB. As can be seen, the eNB MAC, i.e., the file lte-enb-mac.cc, contains the User side of both the CSCHED SAP and the SCHED SAP implementations. The eNB MAC can be used with different schedulers with no modifications including RoundRobin as an example in the figure. The Round Robin scheduler implements the Provider side of the SCHED SAP and
CSCHED SAP interfaces to communicate with the MAC of the eNB. We also take a similar approach to implement other schedulers.

The source files rr-ff-mac-scheduler.{h,cc} files and all other schedulers' files live in the src/lte/model directory. To implement a new scheduler, we need to create its corresponding files and also register it in ns-3. In this project, we have implemented a number of schedulers including 3 schedulers based on the Max Weight scheduling policy. In the next chapter, the details of these schedulers are explained. In this chapter, we discuss how to implement the MW scheduler, as an example, in ns-3. MW scheduler is one of the 3 variations of the Max Weight scheduling policy which will be introduced in Chapter 5 in detail. In brief, the priority metric for MW is the product of queue length and achievable data rate. As mentioned before, mw-ff-mac-scheduler.{h,cc} are the main files which need to be created in the src/lte/model directory. Besides, to register MW as a new scheduler, we should add it to the modulegen__gcc_ILP32.py and the modulegen__gcc_LP64.py files which live in the src/lte/bindings. By rebuilding the project, the wscript in src/lte is regenerated and the new scheduler will be added to the list of the registered schedulers in the file.

The main changes in MW compared to other schedulers are in the DoSchedDlTriggerReq() function. This function is called by the SchedDlTriggerReq() method of the MWSchedulerMemberSchedSapProvider class on its attribute, which is an instance of scheduler type. The main purpose of the DoSchedDlTriggerReq() function is to trigger the scheduling of a DL
5.1.1 Accessing and Updating Queue Size

In `DoSchedDlTriggerReq()` function, the allocation of RBGs is done in nested loops where, for each RBG, the UE who can achieve the highest metric is chosen. Since the MW scheduler metric is the product of a UE queue size and maximum achievable rate, the queue size must be obtained by the scheduler and updated after each RBG allocation. To access a UE queue size, we use the `rlcBufferReq` map which gives the `rlcTransmissionQueueSize` of each user. Note that since we have no idea of the final number of allocated RBGs to each user in the allocation block, we define a local map, named `myMap`, which is a clone of the `rlcBufferReq` map. Algorithm 1 shows how the UE queue size is obtained from `myMap` using the UE’s RNTI.

**Algorithm 1** Obtaining a flow’s queue size

▷ Create `myMap` as a clone of `rlcBufferRequest` map `myMap`

\[
\text{myMap.addAll(rlcBufferRequest.getAll())}
\]

▷ find the corresponding member in `myMap` 

\[
m = \text{myMap.get(flowi)}
\]

\[
queueSize = m.\text{rlcTransmissionQueueSize}
\]

To update the queue size after each RBG allocation, we need to keep track of the history of previous allocations for each UE. This is because the transport block size of a UE, which is determined by the MCS and the number of allocated RBs, increases slightly faster than linearly with the number of allocated RBs to the UE. Therefore, `sizeHistory` is a map defined to update the queue sizes of UEs. Algorithm 2 displays how the `sizeHistory` map is created and used to keep track of the queue sizes.
Algorithm 2 Updating a flow’s queue size

$n =$ number of allocated RBGs to flow $i$

$\triangleright$ find the number of bytes can be transmitted for the flow $i$

$\text{bytesTxed} = \text{AMC}(i, n)$

$\triangleright$ find the corresponding member in myMap

$m = \text{myMap}.\text{get}(i)$

if $(m.\text{rlcTransmissionQueueSize} > 0)$ then

$\triangleright$ find the last change on the flow’s queue size

$\text{lastChange} = \text{sizeHistory}.\text{get}(m)$

if $(\text{lastChange} = \text{Nil})$ then

$\text{sizeHistory}.\text{insert}(m, 0)$

else

$\triangleright$ Revert the last change on the flow’s queue size

$m.\text{rlcTransmissionQueueSize}+ = \text{lastChange}$

end if

$\text{min} = \text{Minimum}(\text{bytesTxed}, m.\text{rlcTransmissionQueueSize})$

$m.\text{rlcTransmissionQueueSize}− = \text{min}$

$\text{sizeHistory}.\text{get}(m) = \text{min}$

end if

5.1.2 Accessing CQI

To access CQI values, we used the $\text{ma30CqiRxed}$ map which gives the sub-band CQI for an individual flow. In detail, this map is filled using the channel quality information of a single flow on a particular RBG in previous TTIs. This approach taken to access CQI in our implemented scheduler is similar to existing schedulers such as FdMt. As mentioned, the CQI used in our scheduler is the sub-band CQI, based on which the MCS is derived from. Then, in turn, using the MCS and RBG size, the flow achievable data rate on the specific RBG is computed.

5.2 Max Weight Schedulers

We have implemented three variations of the Max Weight scheduling policy, as described below:

1. **MR**: The Max Rate (MR) scheduler aims at maximizing the overall throughput of the eNB. It allocates each RBG to the UE that can achieve the maxi-
mum expected data rate over that RBG in the current TTI. Unlike the built-in FdMt scheduler, MR updates the queue length of the UE that gets allocated a new RBG during a TTI, and also does not allocate any RBG to a UE with nothing to send. Let $R_j(k, t)$ denote the maximum rate achievable by UE $j$ on RBG $k$ at TTI $t$. Then, the scheduling decision is made as follows:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} R_j(k, t),$$

where, $N$ denotes the number of UEs in the cell and $\hat{i}_k(t)$ is the UE chosen for transmission on RBG $k$ at TTI $t$.

2. **MW**: This version considers both the queue length and channel quality of each UE. The priority metric for this scheduler is the product of queue length and achievable data rate. As for MR, the queue length of the UE allocated a new RBG is updated and no RBG is allocated to a UE with an empty queue. The scheduling decision for MW is performed as follows:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} Q_j(t) \cdot R_j(k, t),$$

where, $Q_j(t)$ denotes the queue size of UE $j$ before allocating RBG $k$ at TTI $t$.

3. **MQ**: The Max Queue (MQ) scheduler uses UE queue length at eNB as its priority metric. MQ simply sorts UEs in decreasing order of their queue lengths. Therefore, the UE which has more data to send has the highest priority for resource allocation.

Our implemented schedulers are compared against the traditional PF scheduler, as well as FdMt (built-in ns-3 scheduler aiming to maximize throughput) and RR schedulers. The PF scheduler uses the following policy to schedule users:

$$\hat{i}_k(t) = \arg \max_{1 \leq j \leq N} \left( R_j(k, t) / T_j(t) \right),$$

61
where, $T_j(t)$ is the past throughput achieved by UE $j$ until TTI $t$. To calculate $T_j(t)$, a moving average is used as expressed below:

$$T_j(t) = \beta \cdot T_j(t-1) + (1 - \beta) \cdot R_j(k,t),$$

where, $\beta$ ($0 \leq \beta \leq 1$) is the weight factor for the moving average.

### 5.3 Proposed Scheduling Algorithms

In this section, the two proposed schedulers are described in detail:

#### 5.3.1 Q-MW Scheduler

The simulation results of the existing schedulers are presented in the first part of Chapter 6. Analysis of these results, specifically the TCP throughput plot (see Fig. 6.2(a)) reveals that there is a gap of about 10% in the total throughput between 5 and 10 user scenarios. However, this gap does not exist for UDP traffic. Note that in both scenarios (TCP and UDP), we are using the same wireless channel trace meaning that the gap can not be attributed to channel fluctuations. Given that MW selects the UE that has the highest product of achievable channel rate (based on the received channel quality indicator) and queue size, we conjecture that the gap is due to some UEs having no data in their buffer for transmission. In this case, the set of UEs that can be selected for transmission is small and hence there is less chance of scheduling users with the highest achievable rates, leading to a smaller throughput. Indeed, looking at a snapshot of buffer occupancy at eNB for MW (see Fig. 6.8) shows that there are times when the buffer occupancy is extremely low. This situation does not happen with UDP traffic.

To remedy this problem, we need to somehow make sure the UE with small queue gets timely ACKs to increase its sending rate and consequently build up its queue quickly. Therefore, we propose a dynamic scheduler called Q-MW that gives higher priority to UEs with small buffers. Specifically, with Q-MW, the UEs whose queues are below a threshold have
higher priority and are scheduled based on their channel quality only. But UEs whose queue is larger than the queue threshold are scheduled using the MW scheduling policy.

**Q-MW Algorithm**

Q-MW divides UEs into two groups: (a) UEs whose queue is smaller than a threshold $q$, and (b) those whose queue is larger than $q$. The UEs in the first group have higher priority for scheduling over UEs in the other group. The priority metric used for group (a) is as same as MR scheduler while UEs in group (b) are scheduled using MW algorithm. The scheduler assigns RB $k$ at TTI $t$ to UE $\hat{i}_k(t)$, which is selected as follows:

$$
\hat{i}_k(t) = \begin{cases} 
\arg \max_{1 \leq j \leq N} R_j(k,t) & \text{if } Q_j(t) \leq q \\
\arg \max_{1 \leq j \leq N} Q_j(t) \cdot R_j(k,t) & \text{if } Q_j(t) > q 
\end{cases}
$$

where $q$ denotes the threshold on the queue size. The details of the scheduling algorithm are presented in Algorithm 3. Note that the threshold parameter $q$ is an input to the algorithm. In the next subsection, we discuss how $q$ can be selected for a given network configuration in order to improve the throughput of Q-MW.

**5.3.2 H-MW Scheduler**

In the optimal resource allocation problem, the goal is to jointly maximize the sum of the data rates for all users which can be formulated as:

$$
P : \text{Max} \sum_{1 \leq j \leq n} \sum_{1 \leq k \leq m} a_{jk}R_j(k)$$

Subject to

$$
\sum_{1 \leq k \leq m} a_{jk} = 1, \text{ for any } j
$$

$$
a_{jk} = \begin{cases} 
1 & \text{if subcarrier } k \text{ is allocated to user } j \\
0 & \text{if subcarrier } k \text{ is not allocated to user } j 
\end{cases}
$$
Algorithm 3 Q-MW(q) Scheduler

▷ allocate all RBGs to flows

for each RBG $i$ do
  $MW_a \leftarrow 0$
  $MW_b \leftarrow 0$
  $\text{flag} \leftarrow \text{false}$
  ▷ find the flow that has max weight for this RBG
  for each flow $j$ do
    $\text{rate} = \text{AMC}(\text{flow } j, \text{RBG } i)$
    $\text{queue} = \text{QueueSize}(\text{flow } j)$
    if ($\text{queue} > 0$ AND $\text{queue} \leq q$) then
      $\text{weight} = \text{rate}$
      if ($\text{weight} > MW_a$) then
        $MW_a = \text{weight}$
        $\text{mwFlow}_a = j$
      end if
      $\text{flag} = \text{true}$
    else
      $\text{weight} = \text{rate} \times \text{queue}$
      if ($\text{weight} > MW_b$) then
        $MW_b = \text{weight}$
        $\text{mwFlow}_b = j$
      end if
    end if
  end for
  ▷ allocate the RBG to the max weight flow
  if ($\text{flag} = \text{true}$) then
    $\text{mwFlow} = \text{mwFlow}_a$
  else
    $\text{mwFlow} = \text{mwFlow}_b$
  end if
  allocate RBG $i$ to flow $\text{mwFlow}$
end for
It is assumed there are $n$ active TCP flows, and for flow $j$ the rate on subcarrier $k$ is $R_j(k)$. However, due to this problem’s high complexity, a suboptimal solution is proposed in which the assignment is performed in two stages. In the first stage, each RBG is assigned to the user who can support the highest data rate. In the second stage, a MCS which works for all the subcarriers allocated to a user is determined for each user. Although, determining the best MCS values for various subcarriers leads to a higher throughput, in practice today’s algorithms pick the worst CQI to define the associated MCS for each user because of hardware limitations.

For example in Fig. 5.2, let’s assume 6 RBGs are allocated to a particular user. These RBGs can support different data rates but the worst CQI is chosen for the user to transmit data which is the CQI of RBG 6. Based on our observations, there are some cases whereby removing the RBG with the worst CQI from the list of RBGs allocated to a user, increases the overall achievable data rate for the user. As can be seen in Fig. 5.2 the area supported by the first 5 RBGs, which demonstrates supported data rate for the user, is more than the area supported with all 6 RBGs.

Based on this observation, we have developed a simple heuristic in which the RBG with the least CQI will be left unassigned, provided that the achievable data rate of the associated user is improved in the current TTI. This heuristic adds a negligible amount of overhead to the procedure as can be seen in the Algorithm 4 as well.
Algorithm 4 H-MW scheduler

\[ n = \text{number of allocated RBGs to flow } i \]

\( \triangleright \) find the number of bytes can be transmitted for the flow \( i \) using all the allocated RBGs

\[ \text{bytes}Txed_a = \text{AMC}(i, n, \text{GetMcsFromCqi(worstCqi)}) \]

\( \triangleright \) find the number of bytes can be transmitted for the flow \( i \) leaving the RBG \( j \) which has the least CQI unassigned

\[ \text{bytes}Txed_b = \text{AMC}(i, n - 1, \text{GetMcsFromCqi(SecondWorstCqi)}) \]

\[ \text{bytes}Txed = 0 \]

if \( \text{bytes}Txed_a < \text{bytes}Txed_b \) then

\[ \text{bytes}Txed = \text{bytes}Txed_b \]

\( \triangleright \) Remove RBG \( j \) from the map of allocated RBGs to flow \( i \)

\[ \text{rbgMap.at}(j) = \text{false} \]

\( \triangleright \) Set the appropriate mcs for flow \( i \)

\[ \text{mcs}_i = \text{GetMcsFromCqi(SecondWorstCqi)} \]

else

\[ \text{bytes}Txed = \text{bytes}Txed_a \]

\[ \text{mcs}_i = \text{GetMcsFromCqi(worstCqi)} \]

end if
Chapter 6

Results and Discussion

As mentioned in Chapter 1, the work in this thesis is divided into three main parts; in the first part, the performance of existing schedulers in a single cell network is studied. In the second part, two new schedulers, namely Q-MW and H-MW are proposed. And in the last part, effect of handover on the performance of scheduling algorithms in LTE networks is investigated in multi cell networks. In this chapter, we first describe the network configuration and system parameters that have been considered in each of single cell and multi cell networks. Then, the simulation results of each of the three parts are presented.

6.1 Single-Cell Scenario

6.1.1 Network Topology

We have implemented an LTE-EPC program in ns-3 where a remote host sends packets to a number of UEs. The UEs are distributed randomly in the cell within a distance of 500 m to 5000 m from the eNB (see Fig. 6.1). We use channel fading traces for pedestrian mobility (3 km/h speed) that come with the LENA package [20].

![Figure 6.1: LTE-EPC network topology simulated in ns-3.](image)
Table 6.1: System parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC mode</td>
<td>PiroEW2010 [20]</td>
</tr>
<tr>
<td>Fading model</td>
<td>trace-based Pedestrian model</td>
</tr>
<tr>
<td>RLC mode</td>
<td>UM mode</td>
</tr>
<tr>
<td>Tx power of eNB</td>
<td>30 dBm</td>
</tr>
<tr>
<td>Tx power of UE</td>
<td>23 dBm</td>
</tr>
<tr>
<td>Noise figure</td>
<td>5 dB</td>
</tr>
<tr>
<td>Internet delay</td>
<td>10 ms, 30 ms, 50 ms, 100 ms</td>
</tr>
<tr>
<td>Number of UEs</td>
<td>5, 10, 20</td>
</tr>
<tr>
<td>UE locations</td>
<td>random distance from 500 m to 5000 m</td>
</tr>
<tr>
<td>Packet size</td>
<td>512 and 1024 bytes</td>
</tr>
<tr>
<td>Buffer size at eNB</td>
<td>10, 50, 100, 1000 packet</td>
</tr>
<tr>
<td>Simulation time</td>
<td>15 seconds</td>
</tr>
</tbody>
</table>

The remote host is connected to the gateway node of the LTE network with a high-speed link (100 Gb/s) in order to avoid any bottleneck effects outside the LTE network. All UEs are connected to the remote host, while multiple TCP/UDP servers run on the remote host, each server is dedicated to one UE. In fact, there is one TCP/UDP flow from the remote host to each UE. Each flow of traffic is generated by remote host and passes through the gateway to the eNB. The eNB maintains a queue for each flow where traffic flow awaits transmission to associated UE. A scheduler in eNB allocates radio resources to flows by following a specific priority metric (i.e., scheduling policy).

6.1.2 System Parameters

Various system parameters are summarized in Table 6.2. Those parameters that are not listed in the table are used with their ns-3 default values. As can be seen, some of the parameters that deal with the LTE network setup are fixed during the simulations while those that represent network properties, e.g., Internet delay, vary over a wide range of different values. In this work, we are specifically interested in studying the impact of internet delay, eNB buffer size, and packet size on the performance of TCP/UDP under various scheduling algorithms:
6.1.3 Fairness Metrics

To compare the schedulers in terms of fairness, two commonly used metrics, namely $\alpha$-fairness and Jain’s fairness, are used. In $\alpha$-fairness, $\alpha$ is a trade-off factor between fairness and throughput where $\alpha = 0$ gives complete weight to throughput and $\alpha = \infty$ gives complete weight to fairness. A popular choice is $\alpha = 1$, which results in proportional fairness [66]. A scheduler that is proportionally fair, such as PF, should yield the highest metric under $\alpha$-fairness with $\alpha = 1$. A scheduler is proportionally fair if it maximizes the following metric:

$$\sum_{1 \leq i \leq N} \log(x_i),$$

where $x_i$ is the throughput achieved by UE $i$.

Jain’s fairness index is measured by the following metric [28]:

$$\text{Jain’s fairness index} = \frac{(\sum_{1 \leq i \leq N} x_i)^2}{N \cdot \sum_{1 \leq i \leq N} x_i^2}.$$

The maximum value for Jain’s index is 1, which is attained when all UEs achieve equal throughput.

6.2 Multi-Cell Scenario

In the multi-cell network, 3 tri-sectored macrocell sites where each site has 3 cells, i.e., 9 cells in total, are deployed in a hexagonal layout with 500 m inter-site distance. UEs are randomly distributed around the sites and roam the simulation environment with different movement speeds depending on the selected trace-based channel fading model. We use channel fading traces for pedestrian, urban and vehicular mobilities. Similar to single-cell network, in multi-cell topology, the remote host is connected to the gateway node of the LTE network with a high-speed link (100 Gb/s) in order to avoid any bottleneck effects outside the LTE network. All UEs are connected to the remote host, while multiple TCP servers run on the remote host, each server is dedicated to one UE. In fact, there is one TCP flow
from the remote host to each UE. Each flow of traffic is generated by remote host and passes through the gateway to the eNB. The eNB maintains a queue for each flow where traffic flow awaits transmission to associated UE. A scheduler at eNB allocates radio resources to flows by following a specific priority metric (i.e., scheduling policy).

### 6.2.1 System Parameters

Various system parameters are summarized in Table 6.2. Those parameters that are not listed in the table are used with their ns-3 default values. As can be seen, some of the parameters that deal with the LTE network setup are fixed during the simulations while those that represent network properties, e.g., Internet delay, vary over a wide range of different values. In this work, we are specifically interested in studying the impact of handover on the performance of various scheduling algorithms across a wide range of network scenarios in terms of internet delay, eNB buffer size, and fading model parameters.

### 6.3 System Parameters definition

Before going through the simulation results, the definitions of the system parameters used to evaluate the results are presented:

- **Internet delay**: This is the one-way propagation delay between the remote host and the LTE gateway. Using real Internet measurements, it has been reported

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMC mode</td>
<td>PiroEW2010 [20]</td>
</tr>
<tr>
<td>Fading model</td>
<td>Pedestrian, Urban, Vehicular</td>
</tr>
<tr>
<td>Mobility model</td>
<td>Steady-State Random Waypoint</td>
</tr>
<tr>
<td>RLC mode</td>
<td>UM mode</td>
</tr>
<tr>
<td>Packet size</td>
<td>1024 bytes</td>
</tr>
<tr>
<td>Internet delay</td>
<td>10 ms, 20 ms, 50 ms</td>
</tr>
<tr>
<td>Number of UEs</td>
<td>5 in single-cell and 19 in multi-cell</td>
</tr>
<tr>
<td>Buffer size at eNB</td>
<td>10, 50, 100 packet</td>
</tr>
<tr>
<td>Simulation time</td>
<td>15 seconds</td>
</tr>
</tbody>
</table>
that a one-way delay of around 10 ms is representative of Internet delays [13].

Thus, we set the default Internet delay in our experiments to this value, i.e., 10 ms, but will run our experiments with a range of Internet delays as well.

- **eNB buffer size:** Some schedulers take into consideration the user queue size when making scheduling decisions. Moreover, packet loss and delay that affect TCP throughput are highly dependent on the queue size. We consider a range of queue sizes covering small and large queues.

- **Packet size:** This is the maximum segment size of TCP/UDP, which is set to 512 Bytes or 1024 Bytes. The default value is 1024 Bytes.

- **Fading model:** Traces for three different mobility scenarios are provided in ns-3: Pedestrian with mobility speed of 3 kmph, Vehicular with mobility speed of 60 kmph and Urban with mobility speed of 3 kmph.

### 6.4 TCP Throughput with Existing Schedulers

In this work, we study the performance of different scheduling algorithms in terms of their achieved throughput and fairness [57]. While our focus is on Max Weight scheduling algorithms, we provide simulation results for other schedulers for comparison purposes.

Most plots in this section represent the total throughput achieved in the cell by all UEs. For these set of plots, the 95% confidence intervals are also plotted as error bars for each plot. Each individual experiment result is the average of 10 independent simulation runs, each lasting for 15 seconds. To determine the number of simulation runs, we ran the simulations for a number of times and concluded that 10 is a reasonable number since it results in small error bars. The default values for simulation parameters are summarized in Table 6.3.
Table 6.3: Default configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet size</td>
<td>1024 Bytes</td>
</tr>
<tr>
<td>Buffer size</td>
<td>100 Packets</td>
</tr>
<tr>
<td>Internet delay</td>
<td>10 ms</td>
</tr>
</tbody>
</table>

6.4.1 Experimental Design

Two sets of experiments are performed to understand the throughput and delay performance of different scheduling algorithms:

1. *Long-lived flows*: Bulk data transfers of unlimited size are simulated with both TCP and UDP. The goal is to measure the performance of scheduling algorithms in terms of the total throughput and fairness achieved. For the UDP traffic, we allow the remote host to saturate the Internet connection by sending packets back-to-back to the receiving UE.

2. *Short-lived flows*: Motivated by the popularity of web traffic on mobile devices, short TCP flows lasting for only a few round-trip times are also simulated. The goal is to measure the performance of different scheduling algorithms in terms of the time it takes to download objects of various sizes from a remote host.

6.4.2 Long-Lived Saturated Flows

In this scenario, the remote host sends packets to UEs in a back-to-back fashion for the entire duration of the simulation. We are interested in understanding the effect of the number of users, buffer size, packet size and Internet delay on TCP and UDP performance under different schedulers.

*Effect of Number of Users*

In this subsection, we investigate how changing the number of UEs affect the performance of schedulers. Figs. 6.2(a) and 6.2(b) represent the total throughput for saturated TCP
and UDP flows, respectively. In each figure, we compare the total throughput of different schedulers for 5, 10 and 20 UEs. As it can be seen in Fig. 6.2(a), MR and MW, have the highest total throughput among the simulated schedulers for TCP traffic.

An interesting observation is that by increasing the number of UEs, the total TCP throughput increases for all the schedulers. However, for UDP traffic (Fig. 6.2(b)) increasing the number of UEs has almost no effect on the total throughput of the schedulers. The reason for this behavior is that with TCP, if there are only a few UEs in the cell, sometimes the scheduler may not be able to find a user that has both a good channel and data for transmission. With UDP, on the other hand, UEs always have data for transmission and the scheduler is able to find a set of UEs with good channel conditions in every time slot. We will come back to this problem when discussing the effect of buffer size on TCP throughput in subsection 6.6.1.

In Figs. 6.3(a) and 6.3(b), proportional fairness indices of different schedulers are compared. As expected, for both TCP and UDP, PF yields the highest metric. As it can be seen, for small number of UEs, the indices are very close to each other, however, for higher number of UEs, their difference increases. In addition, generally, MW achieves a poor fairness performance compared to other schedulers. In Fig 6.4(a) and 6.4(b) Jain’s fairness indices of the schedulers are compared. We observe that fairness indices of both RR and PF schedulers are very close to each other, while the other two schedulers yield much lower metrics. Again, MW has a low fairness performance, and its performance deteriorates as the number of UEs increases.

Effect of Packet Size

In this subsection, we investigate how changing the packet size affects the performance of different schedulers. The default packet size in our experiments is 1024 bytes. Figs. 6.5(a) and 6.5(b) compare the total throughput for packet sizes 512 bytes and 1024 bytes for TCP and UDP long-lived saturated flows, respectively. We observe that, regardless of the
(a) TCP flows. Two schedulers MR and MW have the highest total throughput. By increasing the number of UEs, the total throughput increases for all schedulers.

(b) UDP flows. FdMt, MR, and MW achieve the highest throughputs. Increasing the number of UEs has almost no effect on the total throughput for all schedulers.

Figure 6.2: Effect of the number of UEs on the total throughput of different schedulers for long-lived TCP and UDP flows.
Figure 6.3: Proportional fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF achieves the highest fairness. For small number of UEs, the indices are very close. In all experiments, MW achieves a poor fairness index.
Figure 6.4: Jain’s fairness index comparison of different schedulers for long-lived TCP and UDP flows. PF and RR achieve the highest fairness while the other two schedulers have yield much lower metrics. In all experiments, MW achieves a poor fairness index and its performance deteriorates as the number of UEs increases.
packet size, MW achieves the highest throughput. Moreover, changing the packet size has no noticeable effect on the TCP total throughput. On the other hand, in the case of UDP, the total throughput for 512 byte packets is slightly less than that of 1024 byte packets. This is expected as smaller packet size means higher header overhead and thus less throughput. However, in the case of TCP, larger packet size may lead to a higher packet loss probability which would negatively affect the TCP throughput. Let $p_b$ and $p$ denote the bit-error-rate (BER) and packet error probability. We have,

$$p = 1 - (1 - p_b)^L \approx 1 - Lp_b,$$

where $L$ denotes the size of the packet. As can be seen, increasing the packet size leads to higher packet error probability. We conjecture that in our experiments the throughput loss due to higher packet error probability cancels out the throughput gain due to a larger packet size.

**Effect of Internet Delay**

Internet delay has a significant effect on TCP performance as it affects the round-trip delay of TCP packets. We change the Internet delay from the default value of 10 ms to 30, 50, 100 ms and measure the total throughput. The results are depicted in Figs. 6.6(a) and 6.6(b). As shown in the figures, while increasing the Internet delay dramatically reduces TCP throughput, it has no effect on UDP throughput. The reason is that when a packet loss happens (e.g., due to buffer overflow), it takes some time for TCP throughput to recover to its highest level, where the recovery time is directly proportional to TCP round-trip time. With UDP, however, the throughput is insensitive to packet losses and Internet delay as it continuously sends packets at a constant rate. Overall, MW achieves the highest throughput for the range of Internet delays considered in this experiment (MW and FdMt results are similar for UDP flows).

1Further investigation is required to make a concrete conclusion. There could be other factors at the lower layers such as packetization, coding, etc that would affect this behavior.
Figure 6.5: Effect of packet size on the performance of different schedulers for long-lived TCP and UDP flows. Regardless of the packet size, MW achieves the highest throughput.
(a) TCP flows. Increasing network delay causes dramatic drop in total throughput for all of the schedulers.

(b) UDP flows. Increasing network delay has no effect on total throughputs.

Figure 6.6: Effect of Internet delay on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput.
Effect of Buffer Size

The default buffer size in our experiments is set to 100 packets (i.e., 100 KByte). Figs. 6.7(a) and 6.7(b) show the performance of different schedulers for a range of buffer sizes. As shown in the figures, increasing the buffer size has no effect on the UDP total throughput. The reason is that the links are in saturation state and thus the buffers are always full regardless of their size. On the other hand, in the case of TCP, as the buffer size increases so does the TCP throughput. The reason is that if there is a drop in TCP rate (e.g., due to a packet loss), with a small buffer at eNB, there is not enough data at the buffer to saturate the wireless link. It is well-known that in order to maximize TCP throughput, a buffer of size equal to the delay-bandwidth product of the link is required. Increasing the buffer beyond the delay-bandwidth product does not result in considerable increase in throughput. As can be seen in Fig. 6.7(a), increasing the buffer size from 100 packets to 1000 packets does not result in any throughput improvement. Interestingly, MW achieves the highest throughput in all experiments.

To see how large a buffer is maintained, we have also computed the average buffer occupancy at the eNB using a moving average estimator. A very large queue would be indicative of TCP gaining throughput at the expense of large delays to the user. On the other hand, if the large buffer is used only for absorbing random fluctuations then the buffer occupancy should be low. Fig. 6.8 shows the moving averages for buffer occupancy (under TCP traffic) for UE 1, 2, 3 and 4 with MW for 100 packets buffer size and 50 ms Internet delay. As shown in the figures, the buffer occupancy of UE 1 suddenly drops and stays very low for a sustained period of time. Note that the plots show the moving averages and not the instantaneous measurements. Having very low queue occupancy for some UEs can result in a throughput loss with TCP, as described earlier in subsection 6.4.2. The throughput loss is more significant when there are only a few UEs in the system as some UEs with good channel may not have any data for transmission in the buffer. In this experiment, there were
(a) TCP flows. TCP throughput increases by increasing the buffer size up to 100 packets. Increasing the buffer size to 1000 packets does not result in any improvement.

(b) UDP flows. Increasing the buffer size has no effect on the throughput as the links are in saturation state regardless of the buffer size.

Figure 6.7: Effect of buffer size on the performance of different schedulers for long-lived TCP and UDP flows. MW achieves the highest throughput in all experiments.
only 5 UEs in the system. Interestingly, a similar behavior was observed for other schedulers as well. This could explain the increase in throughput by increasing the number of UEs that was observed in Fig. 6.2(a).

6.4.3 Short-Lived TCP Flows

In this subsection, we study the performance of different schedulers for TCP short flows. Web traffic on mobile devices is composed of TCP short flows, where each flow is responsible for fetching an object from a remote web server. Unlike long flows which mainly require
high throughput, the major performance bottleneck for short flows is latency or response time, that is, the time it takes to complete the flow (i.e., download the object). To compare different schedulers, the response time for every UE is computed. To compute response time, we measure the time interval from receiving the first packet of a flow until receiving the last packet of the flow at the UE. The average of response times of UEs is the response time for the corresponding scheduler.

To determine the size distribution of short flows, we use the statistics reported in [1] for the popular Internet mail service gmail.com. Based on the measurements in this report, the average size of an HTTP response is 8 KByte. Therefore, in our experiments, we consider 3 web object sizes: small objects (10 KByte), medium objects (50 KByte) and large objects (100 KByte). In each experiment one short TCP flow is simulated and the maximum amount of data to send is set to one of the web object sizes.

Fig. 6.9 shows the response time of different schedulers for different web object sizes. We observe that the response times of RR, PF and MW are very close to each other while FdMt results in slightly longer response time. Overall, there is no significant difference among the simulated schedulers in this experiment.

6.5 TCP Throughput with Proposed Schedulers

In this work, we study the performance of Q-MW and H-MW scheduling algorithms in terms of their achieved throughput. The plots in this section represent the total throughput achieved in the cell by all UEs.

6.5.1 Q-MW Performance Evaluation

Fig. 6.10 represents the throughput gain of MW over PF for different buffer and delay values. As it can be seen, depending on network conditions, sometimes PF performs better than MW. This is contrary to the optimality of MW for inelastic (e.g., UDP) traffic. Overall,
Figure 6.9: Response time comparison for short-lived TCP flows. Overall, there is no significant difference between the response times of different schedulers.

The range of gains varies from $-19\%$ to $38\%$. The throughput gain is computed as follows:

$$\text{gain} = 100\% \times \frac{\text{Throughput}_{\text{MV}} - \text{Throughput}_{\text{PF}}}{\text{Throughput}_{\text{PF}}}.$$

Next, we show the throughput gain of Q-MW over traditional MW and PF. We consider two scenarios where the parameter $q$ is tuned dynamically and statically.

1. Dynamic Tuning In this case, for each combination of buffer/delay, we chose the best threshold $q$ that maximizes the throughput of Q-MW. Currently, we do not have a mechanism for automatically tuning Q-MW, and hence the tuning for these plots is performed manually using a small range of thresholds ($[1, 5]$). Fig. 6.11 shows the throughput gain of the new scheduler for various buffer and delay combinations with respect to traditional MW and PF. It is observed that Q-MW can achieve up to 64% and 37% throughput gain over PF and MW, respectively. On average, the throughput gains over PF and MW are 11% and 10%, respectively. Notice that our goal was to recover the 10% throughput loss of MW when the number of UEs is low.
2. Static Tuning In this case, using off line experiments, we have chosen the thresholds $q = 2, 3, 4$ for eNB buffer sizes 100, 80, 50 packets. These $q$ values result in the highest average gain across the entire range of Internet delays. In practice, the eNB buffer size is a system parameter that is known in advance. Thus, the network operator can choose the best $q$ value, for example, using a lookup table. However, once the $q$ value is set for a buffer size then it does not change over time. In this case, we observed that Q-MW can achieve up to 64% and 29% throughput gain over PF and MW, respectively. On average, the throughput gains over PF and MW are 6% and 7%, respectively. Thus, even a simple static tuning can recover most of the throughput loss of MW.

6.5.2 H-MW Performance Evaluation

In this set of experiments, the number of RBGs is 12. Fig. 6.12 represents the total throughput for 4, 5 and 6 UEs. The average improvement of H-MW compared to MW is %7 which
(a) The gain over PF ranges from $-4\%$ to $64\%$. The average gain is $11\%$.

(b) The gain over MW can be up to $37\%$. The average gain is $10\%$.

Figure 6.11: Throughput gain of Q-MW over PF and MW for different combinations of buffer size and Internet delay.
Figure 6.12: Heuristic scheduler

is obviously more in case of 4 UEs and less for 6 UEs. Since in case of 4 UEs, it is more likely by removing the RBG with the worst CQI from the list of RBGs allocated to a user, the overall achievable data rate for the user improves. Since adding a simple heuristic results in higher total throughput in this experiment, it can be concluded that applying such heuristics with a negligible overhead can be a good solution to improve UEs’ throughputs in real-world scheduling algorithms.

6.6 TCP Throughput with Handover

In this work, we study the performance of PF and MW scheduling algorithms in terms of their achieved throughput. Throughput plots in this section represent the total throughput achieved in the network by all UEs. Since the Each individual experiment result is the average of 10 independent simulation runs, each lasting for 15 seconds. The default values for Internet delay and buffer size are 10 ms and 100 packets, respectively. Since the error bars were really small in all cases, we have not plotted them in this set of experiments.
6.6.1 Single-Cell Network

The remote host sends TCP packets to 5 UEs located in a single cell, in a back-to-back fashion for the entire duration of the simulation where all UEs are connected to a single eNB node.

Effect of Internet Delay

Internet delay has a significant effect on TCP performance as it affects the round-trip delay of TCP packets. We change the Internet delay from the default value of 10 ms to 20 and 50 ms and measure the total throughput. As shown in Fig. 6.13, increasing the Internet delay dramatically reduces TCP throughput. In addition, MW achieves higher throughput in most experiments for the range of Internet delays considered.

Effect of Buffer Size

Fig. 6.14 shows the performance of the schedulers for a range of buffer sizes. It can be seen that as the buffer size increases so does the TCP throughput. The reason is that if there is a drop in TCP rate (e.g., due to a packet loss), with a small buffer at eNB, there is not enough data at the buffer to saturate the wireless link. Again, MW achieves a higher throughput in most experiments.

6.6.2 Multi-Cell Network

The handover algorithm used is the "A2-A4-RSRQ" implemented in ns-3, which selects the target eNB based on the best possible Reference Signal Received Quality (RSRQ). There are 19 UEs in the network that are randomly distributed in the coverage area of 9 cells.

Effect of Internet Delay

For the sake of comparison, we have conducted a set of experiments similar to those conducted for the single-cell network. The results are depicted in Fig. 6.15. Interestingly, unlike the single-cell network, PF achieves higher throughput in most experiments.
Figure 6.13: Effect of Internet delay in a single-cell network: MW achieves higher throughput in most experiments.
Figure 6.14: Effect of buffer size in a single-cell network: MW achieves higher throughput in most experiments.
Figure 6.15: Effect of Internet delay in a multi-cell network: PF outperforms MW in most experiments.
Table 6.4: Average buffer occupancy

<table>
<thead>
<tr>
<th></th>
<th>Pedestrian</th>
<th>Urban</th>
<th>Vehicular</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-Cell</td>
<td>780</td>
<td>3344.67</td>
<td>13884</td>
</tr>
<tr>
<td>Multi-Cell</td>
<td>111.84</td>
<td>716.6</td>
<td>9482</td>
</tr>
</tbody>
</table>

Effect of Buffer Size

The results are depicted in Fig. 6.16. As shown in the figures, contrary to the single-cell network, PF outperforms MW in most experiments.

6.6.3 Discussion

The only change from the single-cell network to the multi-cell network is the introduction of handovers. As alluded to earlier, it is well-known that TCP throughput suffers from the extra latency and packet duplicates caused by handovers. Thus, the reduction in the total throughput in the multi-cell network compared to the single-cell network can be attributed to TCP. However, the inferior performance of MW compared to PF in the multi-cell network is orthogonal to this and can not be explained by TCP behavior.

An interesting observation in both single-cell and multi-cell experiments is that the MW throughput increases as the buffer size increases (see Figs. 6.14 and 6.16). Indeed, the MW algorithm is designed to stabilize user buffers assuming that the buffer occupancy can grow to infinity. If the buffer size is limited (which is the case in any LTE network) then MW may not performed as expected, as can be observed in our experiments. The MW performance is very sensitive to the buffer size and as the buffer size reduces so does the MW performance. To correlate this sensitivity to the behavior we observed in our experiments, we have reported the average measured buffer occupancy in the single-cell and multi-cell network in Table 6.4. As can be seen, the average buffer occupancy with handovers is much lower than that without handovers. While PF is insensitive to buffer occupancy, MW achieves a lower throughput due to lower buffer occupancy.
Figure 6.16: Effect of buffer size a multi-cell network: PF outperforms MW in most experiments.
Chapter 7

Summary and Conclusion

This thesis studied the problem of how scheduling algorithms behave with TCP in LTE networks. To this end, we simulated a number of LTE configurations in ns-3, implemented various packet schedulers, and investigated the performance of ns-3 built-in and implemented schedulers across a wide range of network scenarios in terms of varying parameters.

7.1 Thesis Summary

The work has been done in this thesis, is classified into three main parts:

7.1.1 Study the Performance of Existing Schedulers

In the first part, we studied the performance of existing scheduling algorithms with TCP using real channel traces in LTE networks. We particularly focused on MW scheduling as it has been shown in the literature to be throughput optimal for inelastic traffic. Many works have been so far proposed to generalize MW scheduling however most of them have considered only inelastic traffic. In this thesis, we focused specifically on the impact of resource schedulers on TCP performance using some extra easily available knowledge in LTE networks.

Our simulation results show that MW performs as well or better than other scheduling algorithms including Proportional Fair scheduling (PF). However, we observe that in small cells with only a few users, MW scheduling cannot achieve its full performance compared to larger cells with many users. Our experiments show that MW may result in small queues where some users do not have any data for transmission due to elastic TCP behaviour.
7.1.2 Proposed Schedulers

In the second part of the thesis, two new schedulers were proposed based on MW. The first one, called Q-MW, was developed to remedy the problem observed while studying existing schedulers (Subsection 7.1.1). Based on this observation when there are only a few users in a cell, the throughput of MW drops considerably compared to when there are many users in the cell. Simulation results show that Q-MW can achieve up to 64% and 37% throughput gain, respectively, over traditional PF and MW, if tuned properly.

The second proposed scheduler was not specifically designed for TCP, and is generally more efficient with respect to resource allocation. The main idea behind this new algorithm, named H-MW, is that an RBG allocated to a user which has the worst CQI is left unassigned if the achievable data rate of the associated user is improved in the current TTI. We studied the performance of H-MW and MW schedulers with respect to TCP in terms of the average throughput for a number of users. Our simulation results show that applying this heuristic in MW scheduling algorithm can improve its total throughput by 7% on average.

7.1.3 Study the Effect of Handover on the Performance of Scheduling Algorithms

In the third part of the thesis, we studied the throughput performance of PF and MW schedulers in both single-cell and multi-cell networks. We specifically focused on the impact of handover on these two popular schedulers as the only change from the single-cell network to the multi-cell network is the introduction of handovers.

We found that while MW generally outperforms PF in a single-cell network, in a multi-cell network, PF actually achieves a higher throughput across a diverse set of network configurations in terms of round-trip delay, buffer size and mobility model. Our results show that this behaviour can be attributed to the sudden drops in the occupancy of user buffers at the eNBs due to the hard handover mechanism of LTE, which negatively affects MW as it is sensitive to the amount of backlog in user buffers.
7.2 Future Work

This section discusses complementary interesting areas for future investigation and suggests some avenues to expand the ideas of this thesis.

7.2.1 Automatic Tuning Approach for Q-MW Scheduler

The TCP-aware scheduler proposed in this thesis, namely Q-MW, has a queue threshold parameter that needs to be tuned properly for a given network scenario. However, we currently have no mechanism for automatically tuning Q-MW and it is actually done by running several simulations in terms of the queue threshold parameter to find the best queue threshold. Therefore, designing a dynamic tuning algorithm for Q-MW is left for future work.

7.2.2 Optimized Scheduling for TCP Short Flows

The results presented in this thesis for short-lived TCP flows were entirely related to the performance of some specific schedulers for these flows. However, as nowadays short-lived flows are very popular specially on mobile devices, e.g., cell phone, it is worth to design a scheduling mechanism specifically for this type of flows. In fact, long-flows and short-flows have many different performance bottlenecks which directly affects the implementation of the scheduling algorithm.

Moreover, another interesting area for future research can be designing a TCP mechanism optimized for short-lived TCP flows, as they may require special implementation requirements in different TCP phases. As mentioned earlier, an example for this is the idea to protect short-lived TCP flows in the slow start phase which provides an opportunity for them to increase their congestion window and hence their sending rate.
7.2.3 Power-Efficient Scheduling Algorithms

Another important parameter in designing schedulers is energy consumption which has not been studied in this thesis. This may affect particularly battery driven mobile devices, such as cell phones which should consume as little energy as possible to ensure a long operating time.

The reason behind that is if a scheduling algorithm schedules data for a mobile device in burst then energy consumption will be lower since the mobile device will wake up and receive a large amount of data in high power mode. Hence, it switches to low power mode and remains in this mode for a longer time. On the other hand, if the scheduler is continuously sending a little bit of data, the mobile device has to remain in high power mode to receive the data for a long period of time which is not obviously efficient from energy consumption point of view. Therefore, evaluating schedulers in terms of energy consumption and designing power-efficient scheduling algorithms can be considered as a potential future work.

7.2.4 Generalizing H-MW

The results presented for H-MW are based on removing one RBG from the allocated RBGs to a particular user. It would be worthwhile if the devised algorithm is studied in terms of different parameters. We conducted a set of experiments with the default bandwidth used in the thesis, so it is needed to evaluate the algorithm performance with various values for downlink bandwidth, number of mobile users, and also fading model. For example, intuitively the higher mobility of UEs results in a higher diversity of RBGs’ CQIs; therefore, we can expect to observe more cases where leaving a RBG unassigned leads to a higher data rate for a user in a particular TTI.

The other limitation of our study was the only implemented resource allocation type in ns-3, namely type 0, in which the smallest allocation unit is RBG rather than RB. To generalize H-MW, it is reasonable to assign resources in finer granularity, and so it is needed
to apply other resource allocation types besides type 0.

Moreover, it would be interesting to modify the algorithm and allocate the unassigned resources to other users who can benefit from these resources. Although such a modification adds extra overhead to the algorithm, it would be interesting to investigate how close it is to the optimal resource allocation. Furthermore, removing more than one resource from allocated resources to a user can be investigated as well.
Appendix A

Source Code

In this appendix, the source code of the programs to which we referred in chapter has been provided.

```c
#include "ns3/lte-helper.h"
#include "ns3/epc-helper.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/lte-module.h"
#include "ns3/applications-module.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/config-store.h"

using namespace ns3;

/**
 * Sample simulation script for LTE+EPC. It instantiates
 * a single eNB, attaches one UE to it,
 * starts a flow for the UE to and from a remote host.
 */
NS_LOG_COMPONENT_DEFINE ("LTE/EpcExample");
int
main (int argc, char *argv[])
uint16_t numberOfNodes = 1;
double simTime = 1.1;
double interPacketInterval = 100;

// Command line arguments
CommandLine cmd;

  cmd.AddValue("numberOfNodes",
              "Number of eNodeBs + UE pairs", numberOfNodes);
  cmd.AddValue("simTime", "Total duration of the simulation [s]", simTime);
  cmd.AddValue("interPacketInterval",
              "Inter packet interval [ms]", interPacketInterval);

  cmd.Parse(argc, argv);

Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
Ptr<PointToPointEpcHelper> epcHelper =
    CreateObject<PointToPointEpcHelper> ();
lteHelper->SetEpcHelper (epcHelper);

ConfigStore inputConfig;
inputConfig.ConfigureDefaults();

// parse again so you can override default values from the command line
cmd.Parse(argc, argv);
Ptr<Node> pgw = epcHelper->GetPgwNode ();

// Create a single RemoteHost
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);

InternetStackHelper internet;
internet.Install (remoteHostContainer);

// Create the Internet
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate",
    DataRateValue (DataRate ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
NetDeviceContainer internetDevices =
    p2ph.Install (pgw, remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces =
    ipv4h.Assign (internetDevices);
// interface 0 is localhost, 1 is the p2p device
Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);

Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.
    GetStaticRouting (remoteHost->GetObject<Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo

101
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create(numberOfNodes);
ueNodes.Create(numberOfNodes);

// Install Mobility Model
Ptr<ListPositionAllocator> positionAlloc = CreateObject<ListPositionAllocator>();
positionAlloc->Add(Vector(0, 0, 0));

MobilityHelper mobility;
mobility.SetMobilityModel("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator(positionAlloc);
mobility.Install(enbNodes);
mobility.Install(ueNodes);

// Install LTE Devices to the nodes
NetDeviceContainer enbLteDevs =
    lteHelper->InstallEnbDevice(enbNodes);
NetDeviceContainer ueLteDevs =
    lteHelper->InstallUeDevice(ueNodes);

// Install the IP stack on the UEs
internet.Install(ueNodes);
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address

(Ipv4Address("7.0.0.0"), Ipv4Mask("255.0.0.0"), 1);
(NetDeviceContainer (ueLteDevs));

// Assign IP address to UEs, and install applications

Ptr<Node> ueNode = ueNodes.Get(0);

// Set the default gateway for the UE
Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.
    GetStaticRouting (ueNode->GetObject<Ipv4> ());
ueStaticRouting->SetDefaultRoute
    (epcHelper->GetUeDefaultGatewayAddress (), 1);

// Attach the UE to the eNodeB
lteHelper->Attach (ueLteDevs.Get(0), enbLteDevs.Get(0));

// Install and start applications on the UE and remote host
uint16_t dlPort = 1234;
uint16_t ulPort = 2000;
ApplicationContainer clientApps;
ApplicationContainer serverApps;

PacketSinkHelper dlPacketSinkHelper ("ns3::UdpSocketFactory",
    InetSocketAddress (Ipv4Address::GetAny (), dlPort));
PacketSinkHelper ulPacketSinkHelper ("ns3::UdpSocketFactory",
    InetSocketAddress (Ipv4Address::GetAny (), ulPort));
serverApps.Add (dlPacketSinkHelper.Install (ueNodes.Get(0)));
serverApps.Add (ulPacketSinkHelper.Install (remoteHost));
serverApps.Add (packetSinkHelper.Install (ueNodes.Get(0)));

UdpClientHelper dlClient (ueIpIface.GetAddress (u), dlPort);
dlClient.SetAttribute("Interval", TimeValue
    (MilliSeconds(interPacketInterval)));

dlClient.SetAttribute("MaxPackets", UintegerValue(1000000));

UdpClientHelper ulClient(remoteHostAddr, ulPort);
ulClient.SetAttribute("Interval", TimeValue
    (MilliSeconds(interPacketInterval)));
ulClient.SetAttribute("MaxPackets", UintegerValue(1000000));

clientApps.Add(dlClient.Install(remoteHost));
clientApps.Add(ulClient.Install(ueNodes.Get(0)));

serverApps.Start(Seconds(0.01));
clientApps.Start(Seconds(0.01));

lteHelper->EnableTraces();
// Uncomment to enable PCAP tracing
// p2ph.EnablePcapAll("lena-epc-first");

Simulator::Stop(Seconds(simTime));
Simulator::Run();

Simulator::Destroy();
return 0;
}
Bibliography


[64] TS36.211. Evolved universal terrestrial radio access (e-utra); physical channels and modulation. http://www.etsi.org/deliver/etsi_ts/136200_136299/136211/10.00.00_60/ts_136211v100000p.pdf


Vehicular Technology (VTC), May 2005.

