Experimental Evaluation of Downlink Scheduling Algorithms using OpenAirInterface

Răzvan-Mihai Ursu, Arled Papa, Wolfgang Kellerer
Chair of Communication Networks, Technical University of Munich
razvan.ursu@tum.de, arled.papa@tum.de, wolfgang.kellerer@tum.de

Abstract—**Programmability and softwarization advocate the emerging era of open-source platforms, which embraced by both industry and academia is foreseen as a vital pillar in the construction of next generation mobile networks. Such a valuable open-source project is OpenAirInterface (OAI), which provides a standard compliant mobile network infrastructure, merely based on general purpose hardware computers. While OAI is nowadays widely used by industry and research institutes in proof-of-concept or commercial wireless testbeds, an analysis of the complex functions within the platform is yet to be performed in a large scale. We believe that further research is required to demystify the capabilities of existing tools and present guidelines that alleviate the enhancement and development of additional features. In this context, in this work we shed light on one of the crucial components of any mobile system, namely resource scheduling, while providing an analysis of the available code and instructions to ease the development of new scheduling algorithms based on OAI. Moreover, we demonstrate a performance evaluation of up to 10 UEs for existing and newly implemented scheduling algorithms. Results show, that the development of additional algorithms in OAI is achievable and the experimental behavior follows the theory. Our implementation and observations can serve as a basis for research in the field, and foster the elaboration of theoretical concepts and emerging 5G solutions in practical testbeds.**

Index Terms—**Scheduling, SD-RAN, 5G, OAI, Performance.**

I. Introduction

Programmability and softwarization are foreseen as the main drivers of next generation networks. The ever-growing demand for communication systems providing high reliability, capable of maintaining extremely low latencies and achieving tremendous high throughput, pose significant challenges for existing one-size-fits-all mobile architectures. Therefore, novel solutions such as software-defined networking (SDN) and network slicing (NS) constitute the basis for the development, operation and management of new emerging 5G and beyond mobile network architectures.

In the radio access network (RAN), in particular, softwarization solutions such as SDN referred to as SD-RAN, have been embraced by both industry and academia. Rakuten, Inc. [1] and O-RAN [2] for instance, envision the concept of programmability and virtualization as the main drivers in the 5G/6G era. The concept of open-source programmable code base alleviates the collaboration among industries and fosters immense progress in the mobile community. Additionally, academia plays a major role, as it complements industry with the possibility of researching existing and new developed theories in systems similar to commercial products, narrowing the gap between theory and practice.

OpenAirInterface (OAI) [3] and srsLTE [4] are two examples of academia-based projects, which have gained significant traction in the last couple of years. Both platforms implement the 4G and 5G protocol stack for the user equipment (UE), eNodeB (eNB), gNodeB (gNB), as well as the core network (CN) with ability to operate on general-purpose x86 processors. Leveraging the linux kernel and linux ip protocol stack, together with off-the-shelf software-defined radios (SDRs) [5] they provide a fully fledged mobile network compliant with 3GPP standardization. The development and wide availability of the aforementioned platforms has, in turn, led to the emergence of SD-RAN platforms such as FlexRAN [6] and 5G-EmPOWER [7], that advocate the important concepts of NS in RAN, while providing control over eNBs/gNBs. Both projects are relevant and extremely important for both academia and industry, nonetheless in this work we will further focus our analysis on OAI, as the only system that provides further support with respect to NS and 5G implementation concepts.

As aforementioned, OAI aligned with FlexRAN, support a programmable interface for all the radio access network (RAN) functionalities [1]. Such functionalities vary from statistics collection and monitoring to NS, handover and medium access control (MAC) scheduling to name a few. Although, NS is triggering vast amount of research [8], [9], MAC scheduling still remains one of the most important elements of any mobile system. Utilizing FlexRAN, depending on network conditions, policies can be enforced to the eNBs/gNBs in order to enhance the network capabilities. In that regard, scheduling algorithms that allow for such adjustments become extremely important. However, while there exist a plethora of theoretical papers on MAC scheduling, the practical aspect is often neglected. For instance, currently OAI only provides 3 traditional scheduling algorithms, namely round robin (RR), proportional fair (PF) and maximum channel quality indicator (MT). Given the traction that OAI is receiving, as well as the plethora of works based on this platform, an analysis of existing capabilities and research on extension possibilities becomes extremely interesting.

In this work, we aim at demystifying such concerns and intent to construct a basis for practical research within OAI. Our contributions lie, therefore, in:

[1]https://mosaic5g.io/apidocs/flexran/

1) Diversifying existing traditional scheduling algorithms provided by OAI and extending the range of options by implementing scheduling algorithms that allow adjustments depending on network conditions.
2) Providing, to the best of our knowledge, the first in depth analysis and comparison of scheduling capabilities in the OAI platform, while demonstrating experimentally throughput analysis for up to 10 UEs.
3) Identifying some of the challenges of implementing scheduling algorithms within OAI and providing guidelines for future development.
4) Publishing the full implementation of the new algorithms to further foster experimentation in the field. [2]

The work is structured as follows: Section II provides a broad overview of the most important theoretical aspects of scheduling and mobile network open-source platforms. Section III presents the system design of OAI, as well as the code structure of MAC scheduling algorithms. The design implementation of the new algorithms developed within scope of this work are presented in Section IV. Section V provides an experimental analysis and performance evaluation of all scheduling algorithms within OAI. Finally, the paper is concluded in Section VI while discussing the main findings of this work.

II. Related Work

The development and establishment of OpenAirInterface [3] and srsLTE [4] has triggered vast ongoing research in the mobile community. The aforementioned platforms, provide enormous capabilities and offer the possibility to re-program mobile network functionalities. This has led to the emergence of prototypes in fields such as software-defined radio access networks (SD-RAN) [6], [7], network slicing (NS) [10] and cloud radio access network (C-RAN) [11].

Even though all the aforementioned fields are of high importance to next generation networks, one of the principle functionalities of radio access networks (RANs) remains the MAC scheduling, which in turn constitutes the basis for techniques such as C-RAN and SD-RAN. For instance, state-of-the-art solutions providing NS consider hierarchical scheduling, where within each slice traditional scheduling algorithms are performed [8], [9].

While scheduling has been an actively researched topic from a theoretical point of view [12], [13], [14], practical implementations of large scale in academia were missing until recently. With the development of OAI and srsLTE, an emergence of experimental research has been initiated. However, as observed by [15], both srsLTE and OAI can only provide analysis for a limited amount of UEs, due to software design choices, or expensive hardware equipment. To cater for this issue, in this work we utilize the `oaisim` NFAPI simulator [3] extending experimentation up to 10 UEs.

[2]https://gitlab.eurecom.fr/razvanursu/openairinterface5g-mac-scheduling/-/tree/mosaic5g-oai-sim

[3]https://gitlab.eurecom.fr/mosaic5g/mosaic5g/-/wikis/tutorials/l2-sim

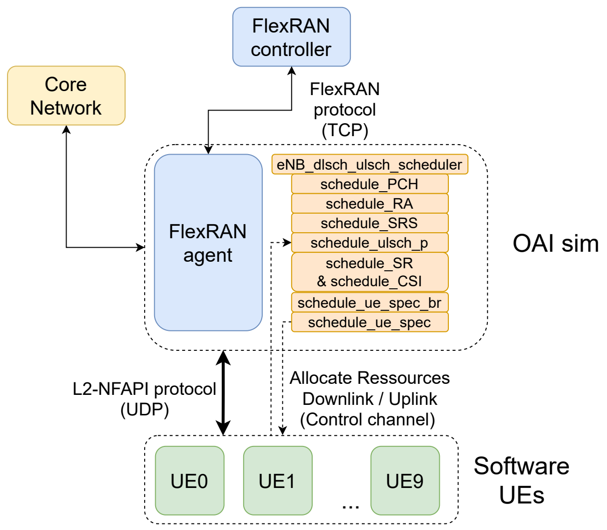


Figure 1: System model architecture based on OAI simulator tool. It contains functionalities within the eNB/gNB with FlexRAN agent related capabilities and in turn connects to the core network, while providing service to UEs using the NFAPI tool.

Works [16], [17], also analyze the performance of OAI while explaining software characteristics. Nonetheless, they mainly focus on evaluating and improving the computational intensity of the functions corresponding to the network and LTE layer-specific calculations.

Similarly to our work, authors in [18] develop a scheduling algorithm for vehicle to everything communications based on OAI. Yet they only provide evaluations for that particular scheduler. Differently, we not only provide guidelines and development details for the additional schedulers, but also demystify the performance of existing algorithms within OAI. To the best of our knowledge, this work provides the largest scale performance analysis of OAI using `oaisim` for up to 10 UEs, considering multiple scheduling algorithms.

III. System Model

In this section we will describe in more detail, the system characteristics of OAI and clarify the functionalities of the blocks that have been modified in order to develop the proposed algorithms.

A. OAI Channel Model

OAI introduces a 3GPP compliant channel model based on a time slotted system, where a slot duration corresponds to 1 ms, known as the transmission time interval (TTI) in LTE. The eNB/gNB serves user equipments (UEs) at each slot. Given an available system bandwidth, a set $\mathcal{R}$ of R physical resource blocks (PRBs) is available, where each PRB has a duration of 0.5 ms and 180 kHz with a subcarrier spacing of 15 kHz. Resources are allocated to UEs on a resource block group (RBG) basis, with a granularity of 2 PRBs. To facilitate the decision making process of schedulers, channel statistics are transmitted towards the eNB/gNB referred to as channel quality

Algorithm 1 WRR

```
UE_id ← 0
Ensure that all UEs in UE_list require at least 1 PRB.
Advance the current RBG, so that it is free.
Set the weights for the UEs.
while UE_list is not empty do
    if there are available RBGs then
        Assign the current RBG to UE UE_id.
        Decrease the number of available RBs by the size of the RBG.
        if UE UE_id does not need PRBs or UE UE_id has already received more PRBs than its weight then
            Eliminate UE_id from the list.
        else
            Increase UE_id.
        end if
    else
        Break;
    end if
    Advance the current RBG so that it points to a free RBG.
end while
```

indicator (CQI). Such messages are retrieved by the physical uplink control channel (PUCCH). Every RBG is associated to a specific CQI value per UE. This is latter mapped to a modulation and coding scheme (MCS), which in turn is translated to a transport block size (TBS). TBS is the total size in bits that can be transmitted by the UE according to 3GPP standardization [19].

OAI can be operated both utilizing real wireless hardware i.e., software-defined radios (SDRs) or in a simulation mode known as `oaisim`, that introduces a physical layer abstraction. Given the fact that scaling the network utilizing SDRs is both expensive and hard to maintain [15], in this work we focus on `oaisim`, where the wireless channel can be emulated by being able to control the CQI value. This allows not only to test our algorithm by still going through the OAI protocol stack, but also fosters reproducibility of the measurements.

B. OAI downlink scheduling

The main components of our system can be observed in Fig. 1. The function `eNB_dlsch_ulsch_scheduler` handles most of the scheduling tasks: it refreshes the list of active UEs and it configures the timers for HARQ synchronous and asynchronous retransmissions. Each of the scheduling algorithms is implemented as part of the `pre_processor.c` and gets called in the `schedule_ue_spec` as `eNB->pre_processor_dl.dl`. This is just a wrapper for a call to the `dlsch_scheduler_pre_processor` function, which retrieves the list of the UEs to be scheduled `UE_to_sched`, the maximum number of UEs that can be scheduled for each TTI, the number of remaining RBGs (`n_rbg_sched`), the RBG allocation mask (which has a 1 if an RBG hasn't yet been assigned and 0 if the RBG has already been assigned) and a pointer to the algorithm's data. Further, the `run` method of the `default_sched_dl_algo_t` class is called, and the pre-processor starts.

After the HARQ retransmissions are served, the buffer size of all UEs are read, and, with the help of the `find_nb_rb_DL` function, the number of needed PRBs (`rb_required[UE_id]`) is calculated, dependenting on the CQI reported for this UE and of the corresponding MCS. After all aforementioned steps, the scheduling algorithm calculates

Algorithm 2 DRR

```
UE_id ← 0
Ensure that all UEs in UE_list require at least 1 PRB.
Advance the current RBG, so that it is free.
Set the quantums for the UEs.
while UE_list is not empty do
    if UE UE_id requires PRBs then
        Increase the deficit_counter of UE UE_id by its quantum.
        while the deficit_counter of UE UE_id $> 1$ and UE UE_id needs resources do
            Assign 1 RBG to UE UE_id.
            Decrease the number of required resources by the number of allocated PRBs.
            Decrease the deficit_counter by the number of allocated PRBs.
        end while
        if the end of list has been reached: then
            Set UE_id to 0.
        else
            Increase UE_id by 1.
        end if
    end if
    if there are no available RBGs then
        Break;
    end if
    Advance the current RBG so that it points to a free RBG.
end while
```

Algorithm 3 PQ

```
UE_id ← 0
Ensure that all UEs in UE_list require at least 1 PRB.
Advance the current RBG, so that it is free.
while there are available RBGs and The UE_list is not empty do
    Find the UE UE_id with the largest buffer size.
    Assign to UE UE_id the minimum between the needed and the available number of RBGs, starting with the current RBG.
    Eliminate the UE_id with the highest buffer size from the list.
    Advance the RBG indicator so that it points to a free RBG.
end while
```

according to the selected policy how many PRBs are to be assigned to each UE, and then the RBG mask is unset for the already allocated RBGs.

The scheduling algorithm is chosen based on the config file of the eNB/gNB, however it can alternatively be changed on runtime with instructions received from the FlexRAN controller through. [4]

IV. Implementation

For the implementation of MAC scheduling algorithms, the `mosaic5g-oai-sim` branch [5] has been utilized. While the round-robin (RR), maximum CQI (MT) and proportional fair (PF) schedulers were already implemented, additionally a weighted round-robin (WRR), deficit round-robin (DRR) and a buffer length based priority queue (PQ) scheduler were implemented within the scope of this work. The main challenge while developing new algorithms remains the time constraint, which demands the scheduling to be performed within 1 ms for 4G and below 1 ms for 5G. That said, for algorithms to present a practical use need to perform in a timely manner.

The algorithms have been implemented using the C programming language and the high level pseudo-codes are presented in Alg. 1, Alg. 2 and Alg. 3, respectively. The WRR portrayed in Alg. 1 works similarly to the classical RR, where a weight w is assigned to each UE, which corresponds to the amount of PRBs that each UE can be allocated. In our case this weight

[4]https://mosaic5g.io/apidocs/flexran/

[5]https://gitlab.eurecom.fr/oai/openairinterface5g/-/tree/mosaic5g-oai-sim

corresponds to 6 PRBS per UE. Once this weight has been reached, the UE will be skipped by the allocation procedure.

Alternatively, DRR shown in Alg. 2 performs based on user-specific quantums. A deficit counter, initialized with 0 and persistent between rounds, as well as a quantum gets associated with each UE. While the list of UEs is not empty and while there are still resources to be allocated, an iteration through all of the UEs is performed and if the UE needs resources, its deficit counter is increased by the quantum value. Once the UE gets served, and for every allocated PRB, the deficit counter decreases by 1. The service for that UE terminates either when all of the required resources have been allocated or when the deficit counter ≤ 1. The algorithm then continues with the next UE and, if the end of the list is reached, the algorithm starts from the beginning in a circular fashion. In our evaluations, the quantums are distributed to UEs as follows: $\{2, 3, 4\}$. For a larger number of UEs, this pattern is repeated.

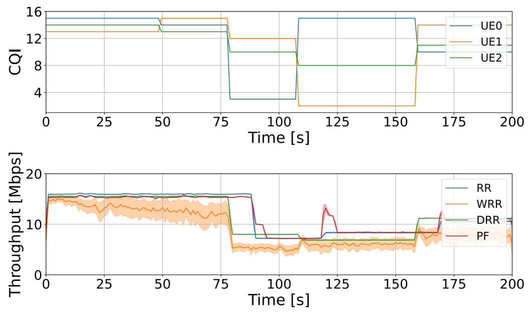
Finally, PQ presented in Alg. 3 assigns resources to UEs based on their current buffer status, while prioritizing UEs whose buffer is larger. In that manner, a certain average buffer size is maintained, aiming at guaranteeing a desired delay. Such an algorithm is especially important for low latency applications.

V. Performance Evaluation

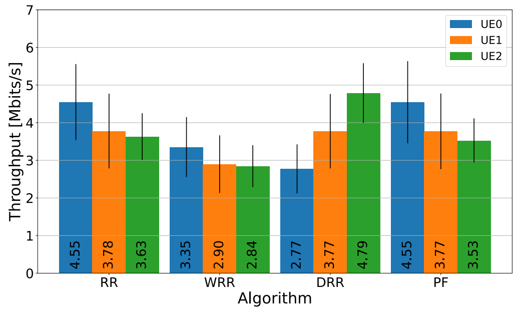
The setup of our performance evaluation is depicted in Fig. 1. Each block corresponds to an OAI entity that can run on a single PC. In our case, the core network runs on separate single PC, whereas the FlexRAN-enabled eNB/gNB and UEs run on the same PC according to the `oaisim` infrastructure.

Apart from the 3 algorithms provided by OAI, the other 3 own developed ones have been evaluated with respect to the achieved throughput while varying the number of UEs between 3, 7 and 10. Through the nature of `oaisim`, the number of UEs is easily scalable and tests can be conducted for, theoretically, up to 255 UEs. The evaluations consist of 200 s, where scheduling is performed in a ms base, whereas average of 1 s are recorded. For every measurement point, 20 repetitions have been conducted and confidence intervals of 95% have been drawn in order to capture the effectiveness of the measurements. The algorithms have been divided into 2 groups, namely starvation free i.e., RR, WRR, DRR, PF and starvation algorithms i.e., MT and PQ. In total there are 25 PRBs reserved for scheduling, representing a 5 MHz system. Results in the following are presented considering these two groups separately to provide fairness in their comparison.

The traffic has been generated by `iperf` TCP traffic, with the client running in the core network PC, whereas the server runs on the UE side. For measuring the traffic, all packets are captured using `tcpdump`, and are post-processed to retrieve the measured throughput and perform the analysis. As the throughput is depending on the CQI (higher CQI means better channel quality and more bits transmitter per each RB), with the help of a `telnet` client in `oaisim`, we control the CQI of each user and change its values in a controllable and repeatable manner ranging from 1 to 15 during the 200 s measurements.



(a) Sum throughput and CQI evolution over time for 3 UEs.



(b) Average UE throughput.

Figure 2: Overall throughput evolution for 3 UEs for starvation free algorithms.

Figure 2a presents the CQI evolution that we used in our experiment.

A. *Throughput Analysis Starvation Free Algorithms*

The initial results for our evaluation performance are portrayed in Fig. 2 for 3 UEs. Fig. 2a demonstrates the evolution of the throughput and CQI values as a sum of all UEs, whereas Fig. 2b portrays the individual UE throughputs. In order to avoid visual clutters, only results for the mean throughput of 7 and 10 UEs are presented and depicted in Fig. 3.

From the results portrayed in Fig. 2, we notice that generally all the algorithms follow the evolution of the CQI values. Given that CQI represents the channel quality, that is the main factor that drives the throughput achieved by UEs. In that regard, higher CQI values translate to higher achievable throughputs and vice versa, as correctly captured by the figure.

An interesting observation is reflected in the comparison among RR and PF scheduler. While PF achieves slightly higher throughputs at specific time instances i.e., $t = 80$ s, $t = 100$ s, in general it behaves similarly to RR. The intuition behind this result lies in the implementation specifics of PF within OAI. The average throughput calculation of each UE follows the equation: $thr\_ue[UE\_id] = (1-a) \times thr\_ue[UE\_id] + a \times b$, where UE_id corresponds to the specific UE, a stands for the window size of calculation i.e., 200 ms in our case, whereas b refers to the current TBS size. If we recall the traditional PF scheduler, for every UE, a coefficient is calculated based on the current UE throughput and the average achieved throughput, where the UE with the highest coefficient receives the resources. A similar approach is followed in OAI, with the

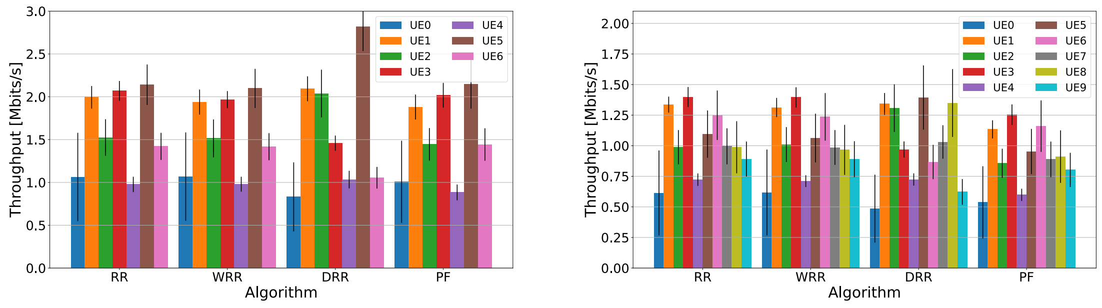


Figure 3: Mean throughput for 7 and 10 individual UEs depicted for various starvation free algorithms.

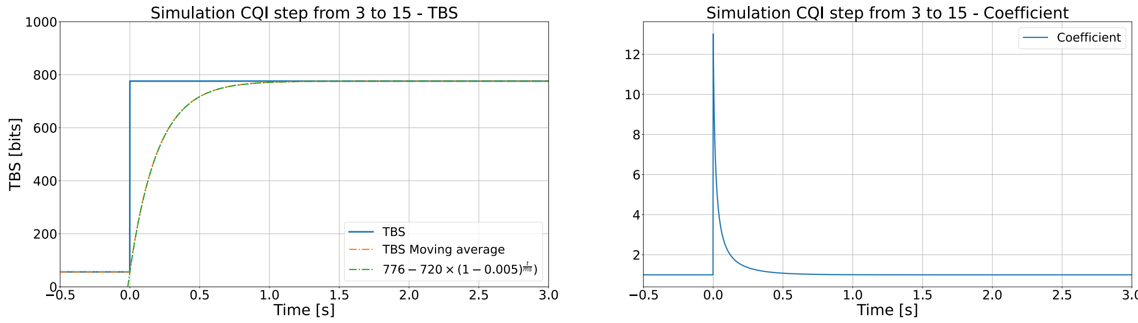


Figure 4: Coefficient calculation evolution for the proportional fair scheduler depending on CQI changes.

difference that the coefficient is calculated as $\frac{b}{thr\_ue[UE\_id]}$. In other words, the coefficient is based on the current achievable TBS and not the achieved throughput. When the CQI values remain constant for a relative long time compared to the time instance of average throughput calculation i.e., 200 ms, that leads to coefficients being equal to 1. In that case, all UEs are treated equally which is as well the case for RR schedulers. Alternatively, when a CQI change occurs, an exponential evolution of the coefficient is observed. For instance, a jump in the CQI from 3 to 15, leads to a jump in the TBS from 56 to 776 bits. Analytically, this leads to an exponential evolution: $thr\_ue(n\Delta t) = 776 - 720 \times \left(\frac{199}{200}\right)^n$, where $\Delta t = 1$ms. A record for this exponential evolution and coefficient is depicted in Fig. 4, which is observed also in throughput results presented in Fig. 2a, where the PF scheduler records a higher throughput.

Furthermore, Fig. 2a illustrates similar behavior for WRR and RR. However, since the weight for the WRR has been set to 6 PRB per UE, with 6 reflecting the maximum PRBs each UE can receive, the overall sum throughput recorded for the RR scheduler is larger given a total of 25 PRBs, compared to 18 PRBs utilized for WRR. This effect is also portrayed with the large variances for the measurements observed with WRR, which nonetheless gets smaller with more UEs as portrayed in Fig. 3, since in that case all 25 PRBs are utilized. Additionally, Fig. 2b shows that for the DRR scheduler the UE with the higher quantums receives more of the resources i.e., UE 3 in our case. Consequently, DRR and RR record a different throughput behavior. Overall, the results demonstrate that the added algorithms behave as expected.

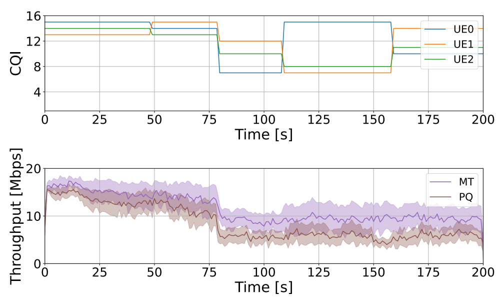
B. *Throughput Analysis Starvation Algorithms*

Similar to the case of starvation free algorithms, results are presented for the starvation algorithms, namely MT and PQ. Results with respect to sum and individual UE throughput for 3 UEs are presented in Fig. 5a and Fig. 5b, whereas results of individual UE throughput for 10 UEs are presented in Fig. 6.

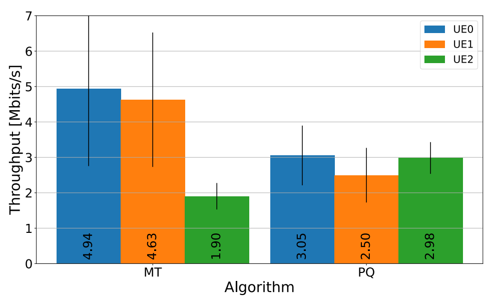
As demonstrated in Fig. 5a, similarly to the starvation free algorithms, the recorded throughput follows the CQI patterns. In general, the MT scheduler outperforms the PQ scheduler as the main objective is to maximize the throughput. On the other hand, the PQ scheduler decides on the resource allocation based on the buffer size. In other words, UEs with the largest buffer size are prioritized for the scheduling. Such kind of scheduler is extremely important for instance for delay critical applications, where each packet delay matters. Given that we assume a full buffer scenario, the priority queue distributes the resources in a more uniform manner, which is even more visible in Fig. 6, where results for more UEs are presented.

VI. Conclusion and Discussion

In this paper we have investigated the potentials of current open-source available platforms in the context of MAC scheduling. By taking OAI as an example, we demystify the capabilities of existing algorithms, while extending the range available schedulers with additional ones. Our results demonstrate compatibility with theoretical models. Moreover, our findings confirm that controlling the resource scheduling can be realized by adjusting weights or quantums, achieving easily configurable throughput performance.



(a) Sum throughput and CQI evolution over time for 3 UEs.



(b) Average UE throughput.

Figure 5: Overall throughput evolution for 3 UEs for starvation algorithms.

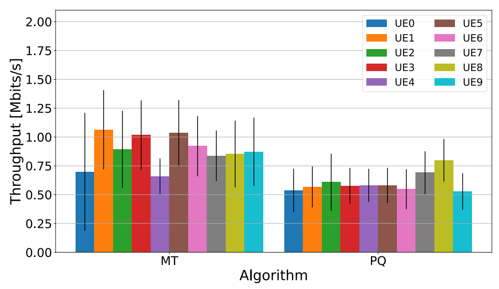


Figure 6: Mean throughput for 10 UEs - starvation algorithms.

While investigating the deployment and performance of scheduling algorithms in OAI, in this section we would like to discuss the decision of implementing scheduling algorithms such as WRR, DRR and PQ. Initially, we would like to emphasize the importance of implementation of various scheduling algorithms in 5G, given the heterogeneous characteristics of applications. That said, specific applications would require distinct schedulers to achieve the optimal performance. Moreover, following the trend of next generation networks towards softwarization and programmability, scheduling algorithms that allow for policy re-configurations with respect to resource allocation, for instance through weights or quantums, depending on network conditions become extremely relevant. Thus, enhancing OAI with the aforementioned schedulers constitutes the basis for further extensions and improvements of OAI leveraging SD-RAN and network slicing.

Acknowledgements

The authors acknowledge the financial support in parts by the Bavarian Ministry of Economic Affairs, Regional Development and Energy as part of the project "5G Testbed Bayern mit Schwerpunktanwendung eHealth" and in parts by the Federal Ministry of Education and Research of Germany (BMBF) in the programme of "Souverän. Digital. Vernetzt." joint project 6G-life, project identification number 16KISK002.

References

[1] Rakuten. (2020) How elegant software can make 5G networks more resilient. [Online]. Available: https://rakuten.today/blog/5g-network-reliability-lightreading.html

[2] O-RAN Alliance e.V. (2019) Operator Defined Open and Intelligent Radio Access Networks. [Online]. Available: https://www.o-ran.org/

[3] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, 2014.

[4] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: an open-source platform for LTE evolution and experimentation," in *Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization*, 2016.

[5] Ettus Research. USRP B210. [Online]. Available: https://www.ettus.com/all-products/ub210-kit/

[6] X. Foukas, N. Nikaein, M. M. Kassem, M. K. Marina, and K. Kontovasilis, "Flexran: A flexible and programmable platform for software-defined radio access networks," in *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*, 2016, pp. 427–441.

[7] E. Coronado, S. N. Khan, and R. Riggio, "5G-EmPOWER: A software-defined networking platform for 5G radio access networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 715–728, 2019.

[8] C.-Y. Chang, N. Nikaein, and T. Spyropoulos, "Radio access network resource slicing for flexible service execution," in *IEEE Conference on Computer Communications Workshops (INFOCOM)*, 2018.

[9] A. Papa, M. Klugel, L. Goratti, T. Rasheed, and W. Kellerer, "Optimizing dynamic RAN slicing in programmable 5G networks," in *IEEE International Conference on Communications (ICC)*, 2019, pp. 1–7.

[10] A. Oliveira and T. Vazão, "Adapting priority schemes to achieve network slice isolation," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 1164–1171.

[11] A. M. Alba, A. Basta, J. H. G. Velásquez, and W. Kellerer, "A realistic coordinated scheduling scheme for the next-generation RAN," in *IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–7.

[12] H. Fattah and C. Leung, "An overview of scheduling algorithms in wireless multimedia networks," *IEEE Wireless Communications*, vol. 9, no. 5, pp. 76–83, 2002.

[13] H. Chaskar and U. Madhow, "Fair scheduling with tunable latency: a round-robin approach," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 592–601, 2003.

[14] J. C. R. Bennett and H. Zhang, "Hierarchical packet fair queueing algorithms," *SIGCOMM Comput. Commun. Rev.*, vol. 26, no. 4, p. 143–156, Aug. 1996. [Online]. Available: https://doi.org/10.1145/248157.248170

[15] F. Gringoli, P. Patras, C. Donato, P. Serrano, and Y. Grunenberger, "Performance assessment of open software platforms for 5G prototyping," *IEEE Wireless Communications*, vol. 25, no. 5, pp. 10–15, 2018.

[16] Y. Y. Chun, M. H. Mokhtar, A. A. A. Rahman, and A. K. Samingan, "Performance study of lte experimental testbed using openairinterface," in *2016 18th International Conference on Advanced Communication Technology (ICACT)*, 2016, pp. 617–622.

[17] A. Virdis, N. Iardella, G. Stea, and D. Sabella, "Performance analysis of openairinterface system emulation," in *2015 3rd International Conference on Future Internet of Things and Cloud*, 2015, pp. 662–669.

[18] J. Manco, G. G. Baños, J. Härri, and M. Sepulcre, "Prototyping V2X Applications in Large-Scale Scenarios using OpenAirInterface," in *2020 IEEE Vehicular Networking Conference (VNC)*. IEEE, 2020, pp. 1–4.

[19] E. Dahlman, S. Parkvall, and J. Skold, *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.