Ingeniería y Universidad
ISSN: 0123-2126
ISSN: 2011-2769
Pontificia Universidad Javeriana

Ariza-Vesga, Luis Felipe; Eslava-Garzón, Johan Sebastián

Real-time Coordinated Scheduling for Cloud Radio Access Networks in a Software-only Environment using the OpenAirInterface Platform

Ingeniería y Universidad, vol. 25, 2021, January-December, pp. 1-19
Pontificia Universidad Javeriana

DOI: https://doi.org/10.11144/Javeriana.iyu25.rcsc

Real-time Coordinated Scheduling for Cloud Radio Access Networks in a Software-only Environment using the OpenAirInterface Platform

Asignación coordinada de recursos en tiempo real para redes de acceso de radio en la nube en un ambiente de solo *software* usando la plataforma OpenAirInterface

Submitted on: September 22, 2019 | Accepted on: August 14, 2020 | Published on: October 12, 2021

Luis Felipe Ariza-Vesga[a]
Universidad Nacional de Colombia, Colombia
https://orcid.org/0000-0003-1262-1413

Johan Sebastián Eslava-Garzón Universidad Nacional de Colombia, Colombia
https://orcid.org/0000-0003-4459-2565

[a] Corresponding author. E-mail: lfarizav@unal.edu.co

doi: https://doi.org/10.11144/Javeriana.iyu25.rcsc

How to cite this article:
L. F. Ariza-Vesga and J. S. Eslava-Garzón, "Real-time coordinated scheduling for cloud radio access networks in a software-only environment using the OpenAirInterface platform," *Ing. Univ.*, vol. 25, 2021 [Online]. https://doi.org/10.11144/Javeriana.iued25.rcsc

Abstract

The objective of this paper is to extend into the OpenAirInterface platform the Coordinated Scheduling (CS) technique to allocate resource blocks among User Equipment (UE) in a wisely way and to control the energy efficiency, the throughput, and the inter-cell interference for Cloud Radio Access Networks (C-RANs). It is achieved by modifying the OpenAirInterface scheduler code, increasing the Remote Radio Unit (RRU) scalability, and employing some component carriers of the Radio Cloud Center (RCC), each one them with one or more UEs. The hardware utilized is composed of general-purpose processors and fast Ethernet transport ports, and the software is recent frequency-domain methodologies in a software-only environment where the use of radio units are not required. However, the USRP B200 mini-i radio unit and the UE (Samsung Galaxy S8) were considered only for validation purposes. The emulations using frequency-domain methodologies, compatible with fourth and fifth-generation cellular systems, allowed real-time emulations and reduced 10-fold the multipath channel's signal processing complexity compared to time-domain methodologies. The results show we can emulate a real-time static coordinated scheduling proof-of-concept for one C-RAN composed of one RCC, three RRUs, and three UEs. In the end, it is evaluated the reproducibility and the scalability of synthetic networks composed of one RRU and at least one UE, without using software-defined radio units, reducing prototyping uncertainties of the physical hardware and the total price of the experiment.

Keywords: Coordinated scheduling, C-RAN, frequency-domain methodologies, OpenAirInterface, software-only environment, synthetic network, time-domain methodologies.

Resumen

El objetivo de este artículo es extender dentro de la plataforma OpenAirInterface la técnica de programación coordinada de recursos para reservar inteligentemente bloques de ellos entre los usuarios móviles y controlar la eficiencia energética, la rata de bits efectiva y la interferencia entre celdas para redes de acceso de radio en la nube (C-RAN en inglés). Esto es llevado a cabo modificando el código de programación de recursos de OpenAirInterface, incrementando la escalabilidad de las unidades de radio remotas (RRU en inglés) y empleando varios componentes de portadoras del centro en la nube de radio (RCC en inglés), cada uno de ellos con uno o más usuarios móviles (UE en inglés). El *hardware* utilizado se compone de procesadores de propósito general, puertos rápidos de transporte Ethernet y el *software* son metodologías en el dominio de la frecuencia recientemente desarrollados en un ambiente de solo *software*, donde el uso de la unidad de radio no es necesario. Sin embargo, la unidad de radio, que en este caso es el USRP B200 mini-i, y el usuario móvil (Samsung Galaxy S8) fueron considerados únicamente para propósitos de validación. Las emulaciones usando las metodologías en el dominio de la frecuencia, compatible para sistemas celulares de cuarta y quinta generación, permitió realizar emulaciones en tiempo real y reducir diez veces la complejidad el procesamiento de señales de canal de múltiples trayectorias comparado con las metodologías en el dominio del tiempo. Los resultados muestran que nosotros podemos emular una prueba de concepto en tiempo real de una programación coordinada y estática de recursos para una C-RAN compuesta por un RCC, tres RRUs y tres UEs. Al final, es evaluada la reproducibilidad y la escalabilidad de las redes sintéticas compuestas por un RRU y al menos un usuario móvil sin usar unidades de radio definidas por *software*, reduciendo las incertidumbres de prototipado del *hardware* físico y el precio total del experimento.

Palabras clave: programación coordinada de recursos, C-RAN, metodologías en el dominio de la frecuencia, OpenAirInterface, ambiente solo de software, redes sintéticas, metodologías en el dominio del tiempo.

Introduction

The fifth generation (5G) of cellular networks is being deployed around the world, and some use cases have been designed, such as the Enhanced Mobile Broadband (eMBB), the Massive Machine Type Communications (mMTC), and the Ultra-Reliable and Low Latency Communications (URLLC). The eMBB use-case will require wider bandwidths to support the increasing amount of traffic or sustain bandwidth-hungry applications such as holographic communications. The next mMTC use case will enable billions of things connected to the internet (IoT) in applications such as smart cities. Finally, the URLLC use case will require low latency to make possible applications such as remote surgery or autonomous driving [1]. New emerging radio technologies such as mmWaves, small cells, massive MIMO, and beamforming have been proposed and adopted to solve issues like traffic saturation, macrocell congestion, capacity, and spectral efficiency.

Another important technology is the Cloud Radio Access Network (C-RAN), which has evolved through some generations of cellular networks. It disaggregates the base station's traditional concept between new entities called the Base Band Unit (BBU) and the Remote Radio Head (RRH). These entities are connected through the CPRI fronthaul interface, and BBUs are pooling at centralized Data Centers (DCs) [2]. However, in massive MIMO scenarios, the fronthaul interface increases its capacity at an unsustainable level directly related to the number of antennas. The new fronthaul and midhaul interfaces are used to solve the previous issue. These interfaces connect new entities called the Radio Cloud Center (RCC), the Radio Aggregation Unit (RAU), and the Remote Radio Unit (RRU). The location of these entities depends on the functional split that redistribute some baseband functions among the RCC, the RAU, and the RRU and allow the fronthaul adaptation between bandwidth and latency. Also, the C-RAN supports complex scenarios and it is built on virtualized network components, general-purpose hardware, and standardized interfaces enabling hardware commoditization [3], [4]. However, mobile network operators still need to embrace the principles of intelligence and openness for flexible, multi-vendor, autonomous, versatile, and efficient cellular ecosystems.

One of the principles of intelligence studied in this paper is the Coordinated Multipoint (CoMP) technique, which improves the radio network cooperation among RRUs with different Component Carriers (CCs). In literature, for downlink transmissions, we found three schemes. The first one is the Coordinated Scheduling (CS), where the evolved Node B (eNB) or the RCC makes scheduling decisions about the allocation of Physical Resource Blocks (PRBs). The second one is the Joint Transmission (JT), where multiple eNBs or RCCs transmit a specific message to UEs located at the cell-edge. Finally, we have the Dynamic Point Selection (DPS) scheme, where a different eNB or RCC serves a UE without employing handover procedures each Time Transmission Interval (TTI) [5]. Recently, the artificial

intelligence was utilized to enable an F-RAN testbed employing the OpenAirInterface (OAI) emulation platform [6].

We focus our study on the CS scheme built on the top of OAI [7], [8]. In that platform, we consider the centralized scheduler behavior and the smart allocation of PRBs. From literature, we found some related works which have employed the OAI platform. The first one is a virtual RAN prototype to pool Base Band Units, which is enforced by a centralized coordinator [9], [10]. The second one is a real-time CS proof-of-concept to analyze the feasibility and its benefits [11]. The third one is a dynamic CS based on a virtual cell control [12]. Finally, the last one is a RAN optimization that employs the FlexRAN platform [13]. All of them use physical Commercial-Off-The-Shelf (COTS) UEs and Remote Radio Units (RRUs), nevertheless as far as we know, nobody has proposed a software-only emulation framework in real-time described for the first time in this paper.

We contribute with a software-only emulation framework to study the real-time static load-unaware CS technique. The PRBs are distributed according to the number of active UEs and CCs. In particular, we can create a hybrid emulation framework to mix both real-time synthetic networks (composed of one RRU and multiple UEs) and Radio Frequency (RF) hardware. In this case, synthetic network components are used for network scalability and real-time RF hardware for application testability. Notwithstanding, it is not the focus of this paper.

Additionally, we contribute with a proof-of-concept to test system-level emulations and evaluate the viability of 3GPP standard-compliant prototyping using not only time-domain but also frequency-domain methodologies. We pool RRUs using the IF4P5 interface (7.1 functional split of the 3GPP standard) where in-phase and quadrature signals are exchanged in the frequency domain and skip the use of radio units to accomplish a software-only emulation. This approach increases the reproducibility, improves the scalability, and reduces prototyping uncertainties of software-defined radio frontends, which is an alternative for people without experience with radio devices or those who do not want to invest in expensive radio units.

The rest of the paper is organized as follows. Second section describes the coordinated scheduling technique, and third section reports the performance evaluation. Finally, in fourth section, we outline conclusions and future works.

Problem Statement and Methodology

C-RANs are evolving through generations of cellular networks and the next goal is to be open and smart. For instance, the O-RAN alliance constituted by vendors, mobile operators, and contributors work together to find a way to control a complex C-RAN for next-generation cellular networks. The intelligence introduced with artificial intelligence is employed to control the network system and obtain less power consumption, higher spectral efficiency, and less inter-cell interference. However, it is necessary to find variables to control it, such as the allocation of PRBs utilizing the CS technology that we explore in this paper.
The methodology that increases the inter-cell interference and validates the CS technique is the following:

1. Use the OAI platform to run system-level emulations. This platform is utilized for many vendors, universities, and mobile operators around the world.
2. Identify the OAI routines that control the scheduler task.
3. Extend the scheduler functions to enable the CS technique.
4. Evaluate and validate the system-level performance of the CS technique for C-RANs.

Coordinated Scheduling

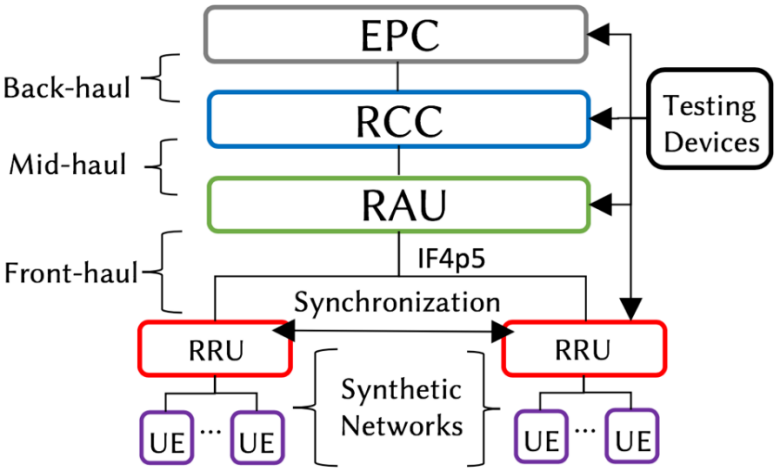
The C-RAN architecture is displayed in figure 1 (throughout the whole paper, we use Next Generation Fronthaul Interface (NGFI) terminologies [2]). The C-RAN has three different network segments called the fronthaul, the midhaul and, the backhaul. The fronthaul connects the RRU to the RAU, the midhaul links the RAU to the RCC, and the backhaul anchors the RCC to the Evolved Packet Core (EPC). The RAU capabilities in the whole article are considered inside the RCC, and a midhaul interface is not required.

Even though the centralization promises performance improvements and cost reductions, it has challenges related to the selection of the fronthaul and the functional split for different applications [14]. We use the IF4P5 interface to take advantage of new frequency-domain methodologies to enable real-time emulations in a software-only environment and to reduce the fronthaul bitrates in both downlink and uplink [14]. It can be handled centralizing those functions benefiting the most from coordination and leaving the others distributed.

The scheduler is the one that exploits the most from centralization. Although, its effectiveness depends on the location of C-RAN layers, such as the Physical (PHY), Medium Access Control (MAC), Radio Link Control (RLC), Packet Data Convergence Protocol (PDCP), Radio Resource Control (RRC), and Internet Protocol (IP). The RAN functions are classified in scheduler-dependent, scheduler-independent, and scheduler-itself. The scheduler-

dependent includes the PHY, MAC, and low-RLC layers, and the scheduler-independent considers the RRC, PDCP, IP, high-RLC layers. The advantages of coordination will rely on the distribution of the PHY, MAC, and low-RLC scheduler-dependent functions [11].

Figure 1. C-RAN architecture considering two synthetic networks, each one composed of one RRU and multiple UEs. It takes advantage of frequency-domain methodologies and simulates de channel model to accomplish real-time emulations without using radio units.



Source: Own source

OpenAirInterface Scheduling Framework

In a Long Term Evolution (LTE) system, the architecture of the air interface is built with a protocol stack as depicted in figure 2. At the transmitter side, the application creates data packets in the user plane processed by IP, User Datagram Protocol (UDP), and Transmission Control Protocol (TCP) [15]. Then, the RRC protocol creates messages to be exchanged between the base station and the UE. Previous control and user plane messages are processed by the PDCP, the RLC, and the MAC, before being passed to the PHY. This last layer manages the transport channel processor to apply error management procedures, the physical channel processor to handle the Orthogonal Frequency Division Multiple Access (OFDMA) scheme, and the analog processor to convert data into analog signals.

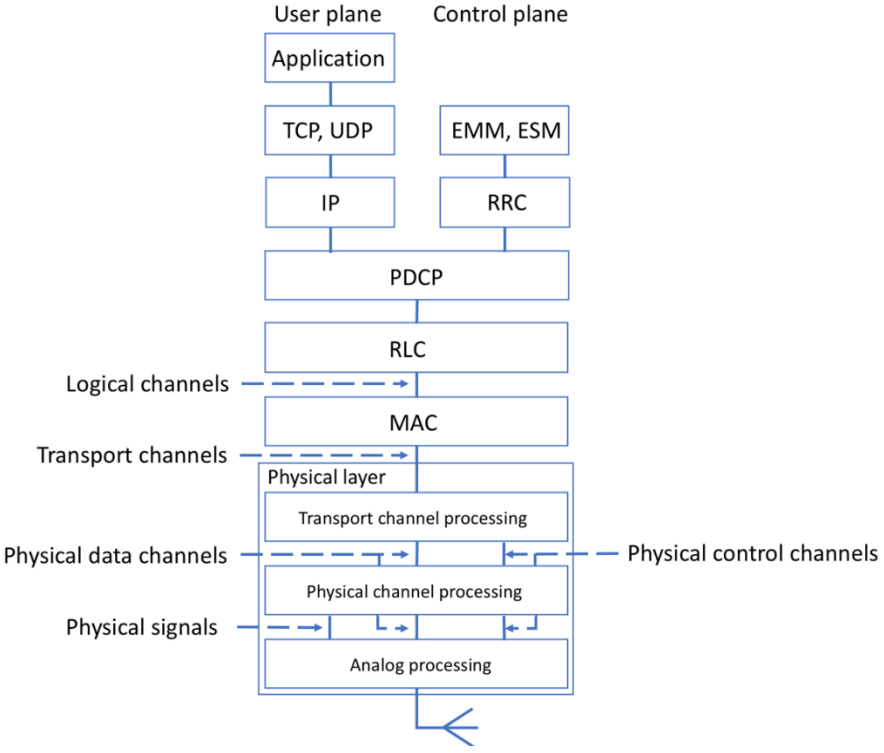
The scheduler is located at the MAC layer, and this layer interacts with the physical layer through transport channels. The essential transport channels are the Uplink Shared Channel (UL-SCH) and the Downlink Shared Channel (DL-SCH) that carry data and signaling messages across the air interface. Among other channels, the Random Access Channel (RACH) is employed by the UE to contact the base station without any scheduling assignation. The remaining channels are the Paging Channel (PCH), the Broadcast Channel

(BCH), and the Multicast Channel (MCH) that manages paging messages, the Master Information Block (MIB), and data from the broadcast/multicast multimedia service, accordingly.

At the MAC layer, OAI implements a stateful scheduler and allocates PRBs every TTI (in this case, the TTI corresponds to 1 ms). It takes decisions across multiple TTIs based on the Channel Quality Information (CQI) reported by UEs. Then it decides the Modulation and Coding Scheme (MCS). The OAI scheduler's primary operations are carried out inside the *eNB_dlsch_ulsch_scheduler* () function called by *schedule_ue_spec* () function, as depicted in figure 3. The latter function is called by the *phy_procedures_eNB_TX* () function for the component carrier 0 (CC0), and then it triggers scheduling for remaining CCs.

Afterwards, the *dlsch_scheduler_pre_processor* () function is called by the *eNB_dlsch_ulsch_scheduler* () function responsible for the scheduler algorithm. There is limited information about the scheduler function. Notwithstanding, in reference [16], we can understand where and how it is implemented in OAI.

Figure 2. Architecture of the air interface protocol stack

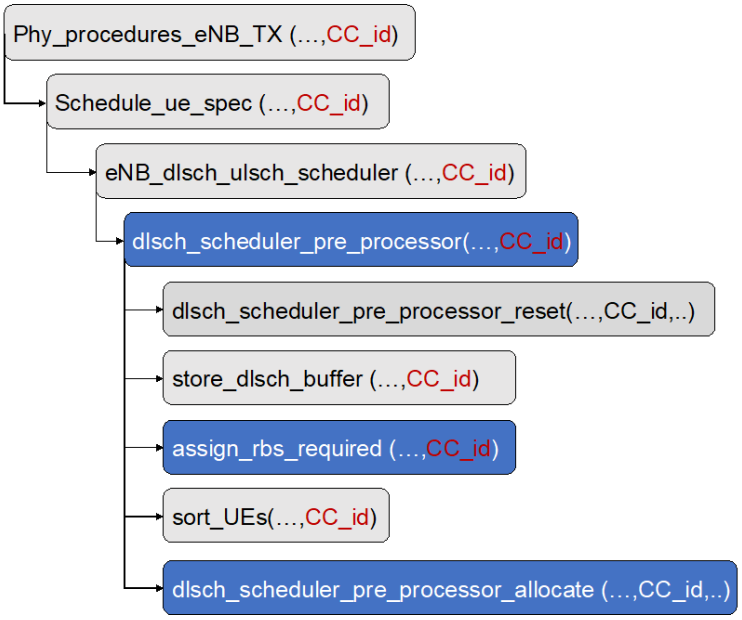


Source: Own source.

Static Load-unaware Coordinated Scheduling

Previously, for the C-RAN, we proposed real-time and 3GPP standard-compliant frequency-domain methodologies to enable the synthetic network scalability in a software-only environment [17]. The synthetic network is composed of one RRU and multiple UEs. These methodologies reduce the average computation time of software functions of the multipath channel 10-fold compared to the time-domain methodologies [17]. The static load-unaware coordinated scheduling approach is considered to evaluate the allocation of PRBs, which are distributed between UEs in a fairly way. Our OAI extension is available in reference [18]. In the following, we discuss CS implementation.

Figure 3. Call graph of the OAI scheduler



Source: Own source.

First, we modified legacy *phy_procedures_eNB_TX* (), *schedule_ue_spec* (), *eNB_dlsch_ulsch_scheduler* (), *dlsch_scheduler_pre_processor* (), *store_dlsch_buffer* (), *assign_rbs_required* (), and *sort_UEs* () functions to introduce a new variable in the parameter list called *CC_id* as depicted in red of figure 3. It allows the scheduler to work every CC at a time. Second, we removed any *for loop* to avoid redundant tasks that depends on the *CC_id* variable. Finally, for the CS algorithm we modified the code in *dlsch_scheduler_pre_processor* (), *assign_rbs_required* (), and *dlsch_scheduler_pre_processor_allocate* () functions (blue boxes in figure 3) as figures 4, 5, and 6 explain in pseudo-code.

Our extension is resumed in 5 steps:

1. Initialize scheduling information and prepare the scheduling data structures for all active UEs.
2. Store the DL-SCH buffer for each logical channel of the RLC layer.
3. Calculate the number of PRBs required by each active UE based on logical channel buffer, the number of active UEs, and the number of CCs (CC_total).
4. Sort the users based on DL-SCH logical channel buffer and CQI.
5. Assign PRBs based on the number of active UEs and the CC_total number. Then compute average PRBs per active UE per CC. Finally, the resource blocks allocation task is done in 2 rounds. The first one assigns the average number of PRBs to each UE and the remaining PRBs to UEs with high priority.

Figure 4. Algorithm 1

Algorithm 1 dlsch_scheduler_pre_processor(..., CC_id)

```
dlsch_scheduler_pre_processor_reset(...,UE_id, CC_id...)
store_dlsch_buffer(Mod_id, frameP, subframeP, CC_id)
assign_rbs_required (Mod_id, frameP, subframeP..., CC_id)
sort_UEs (Mod_id, frameP, subframeP, CC_id)
for (Loop over all active UEs) do
    if (round > 0) then
        if (CC_total < 2) then
            nb_rbs_required[CC_id][UE_id] = nb_rb[harq_pid]
        else
            if (CC_idx==CC_id && N_RB_DL%CC_total!=0) then
                if (nb_rb[harq_pid] > N_RB_DL/CC_total+1) then
                    nb_rb[harq_pid]= N_RB_DL/CC_total+1
                    nb_rbs_required[CC_id][UE_id] = nb_rb[harq_pid]
                else
                    nb_rbs_required[CC_id][UE_id] = nb_rb[harq_pid]
            else
                if (nb_rb[harq_pid] > N_RB_DL/CC_total) then
                    nb_rb[harq_pid]= N_RB_DL/CC_total
                    nb_rbs_required[CC_id][UE_id] = nb_rb[harq_pid]
                else
                    nb_rbs_required[CC_id][UE_id] = nb_rb[harq_pid]
    if (nb_rbs_required[CC_id][UE_id] > 0) then
        total_ue_count+=1
    if (nb_rbs_required[CC_id][UE_id] == 0) then
        nb_rbs_required[CC_id][UE_id] = 0
    else if (min_rb_unit*total_ue_count<=N_RB_DL/CC_total) then
        average_rbs_per_user[CC_id]=N_RB_DL/CC_total
    else
        average_rbs_per_user[CC_id]=min_rb_unit
for (Loop over all active UEs) do
    Control channel or retransmission
for (Loop over 2 rounds) do
    dlsch_scheduler_pre_processor_allocate(...,CC_id,...,subframeP)
```

Source: Own source.

It is essential to mention that we considered multiple CCs that are not adjacent. In this case, we change the Mod_id value of the PHY_VARS_eNB structure to scale RRUs. It means the inter-cell interference is not calculated to improve the real-time validation. Nevertheless, it is not necessary for static coordinated scheduling because there is no overlapped PRBs.

Figure 5. Algorithm 2

Algorithm 2 assign_rbs_required (Mod_id,...,CC_id)

```
for Loop over all active UEs do
    Update CQI information for CC_id
    Provide the list of CCs sorted according to MCS
    if (dl_buffer_total>0) then
        while (TBS < dl_buffer_total) do
            nb_rbs_required[CC_id][UE_id]+=min_rb_unit[CC_id]
            if (CC_total < 2) then
                if (nb_rbs_required[CC_id][UE_id] > N_RB_DL) then
                    TBS = get_TBS_DL(dlsch_mcs1,N_RB_DL)
                    nb_rbs_required[CC_id][UE_id]=N_RB_DL
                    break
            else
                if (CC_idx==CC_id && N_RB_DL%CC_total!=0) then
                    if nb_rbs_required[CC_id][UE_id]>N_RB_DL/CC_total+1 then
                        TBS = get_TBS_DL(dlsch_mcs1,N_RB_DL/CC_total+1)
                        nb_rbs_required[CC_id][UE_id] = N_RB_DL/CC_total+1
                        break
                else
                    if (nb_rbs_required[CC_id][UE_id] > N_RB_DL/CC_total) then
                        TBS = get_TBS_DL(dlsch_mcs1,N_RB_DL/CC_total)
                        nb_rbs_required[CC_id][UE_id] = N_RB_DL/CC_total
                        break
            TBS = get_TBS_DL(dlsch_mcs1,nb_rbs_required[CC_id][UE_id])
```

Source: Own source.

Figure 6. Algorithm 3

Algorithm 3 dlsch_scheduler_pre_processor_allocate(CC_id,...)

```
N_RBG_CS=N_RBG/CC_total
if (CC_total= 1) then
    start=0
    stop=N_RBG
else
    if (CC_idx_last==CC_id && (N_RB_DL%CC_total)!=0) then
        start=CC_id*N_RBG_CS
        stop=(CC_id+1)*N_RBG_CS+1
    else
        start=CC_id*N_RBG_CS
        stop=(CC_id+1)*N_RBG_CS
for i=start to stop do
    Allocation process
```

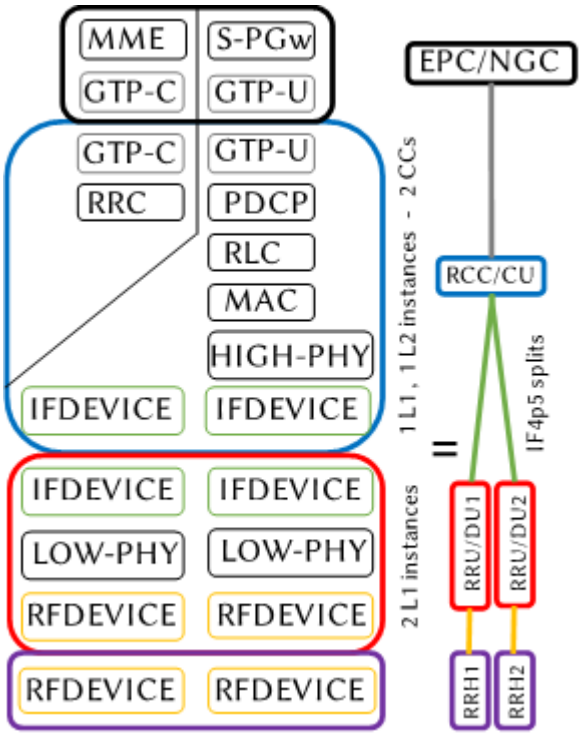
Source: Own source.

Synthetic Networks Scalability

In figure 7, we observe a scenario of two RRUs, two RRHs, one RCC, and one Evolved Packet Core (EPC) for an indoor environment with full centralization at the RCC. The two RRUs have one layer 1 (L1) instance for the low physical layer, respectively. Then, the RCC has one L1 instance for the high physical layer, one layer 2 (L2) instance, and two CCs. Finally, the RCC has two southbound IP-based IFDEVICEs (one per CC). Each RRU has one northbound IP-based IFDEVICE to reach the RCC and one RFDEVICE to reach the RRH. The IFDEVICE is an interface related to the functional split, and the RFDEVICE interface is related to the radio unit (we use one RRH and one RFDEVICE only to validate the allocated PRBs with a USRP, but they are not necessary in software-only emulations accomplished in this paper). We can accommodate three synthetic networks with one UE

each one in a C-RAN to run real-time emulations using frequency-domain methodologies [19]. More significant scenarios are supported, increasing the number of CCs and RRUs, but they do not work in real-time employing generic x86 computers as described in the performance evaluation section. It is recommended to use a High-Performance Computing (HPC) to enable an emulation of more than three synthetic networks in real-time using the same methodologies. Nevertheless, this issue remains as future work.

Figure 7. Synthetic network scalability



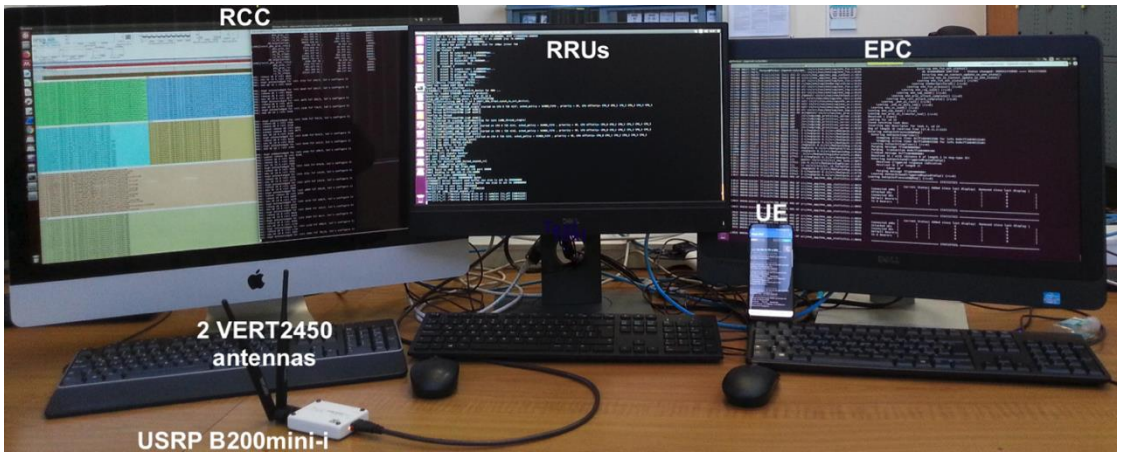
Source: Own source.

It should be mentioned that frequency-domain methodologies can be used to mix both real-time synthetic network components and RF hardware to create a hybrid emulation framework. The former is used for network scalability and the latter for application testability. However, in our prototype, we use synthetic networks in a software-only environment to validate the static load-unaware coordinated scheduling behavior. The next section reports the performance evaluation of the software-only experiment.

Performance Evaluation

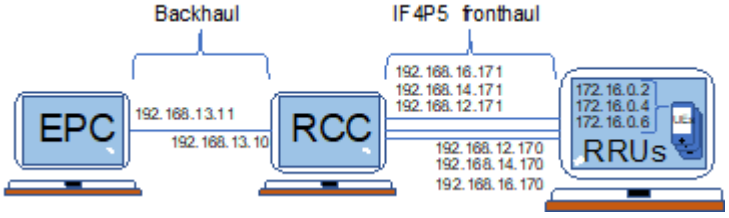
In our software-only emulation prototype, we used an 8th generation Intel Coffee Lake core I7-8700x6 (Ubuntu 16.04, kernel 4.4.0-133-lowlatency) computer to emulate the RRUs, a 2nd generation Intel core I5-2400x2 (Ubuntu 14.04, kernel 4.4.0-31-generic) computer to emulate the RCC, and a 3rd generation Intel core i7-3700x4 (Ubuntu 16.04, kernel 4.13.0-45-generic) computer to emulate the EPC, as shown in figure 8. The USRP B200mini-i and the Samsung Galaxy S8 were utilized only for validation of allocated PRBs when we restrict by software the first and second half of total PRBs for one UE.

Figure 8. Hardware utilized for emulations. It is composed of three computers, one USRP B200mini-i, two antennas, and one COTS UE (Samsung Galaxy S8). The validation of allocated PRBs employs all the hardware, but the performance evaluation for software-only emulations only requires computers



Source: Own source.

Figure 9. The C-RAN scenario is composed of 3 PCs for the EPC, the RCC, and the RRUs



Source: Own source.

The IPv4 C-RAN setup uses 192.168.12.0/24, 192.168.14.0/24, and 192.168.16.0/24 network segments for the fronthaul (each for different synthetic network), and the 192.168.13.0/24 network segment for the backhaul, as depicted in figure 9. The RRUs and UEs' locations are defined at random. How UEs are attached to the network through any RRU is selected by the lower path loss. The number of PRBs allocated does not depend on the location of UEs (the user throughput is related to the quality of the channel reported by

themselves), but they are correlated to the traffic. For all the experiments we use the emulation parameters presented in Table 1.

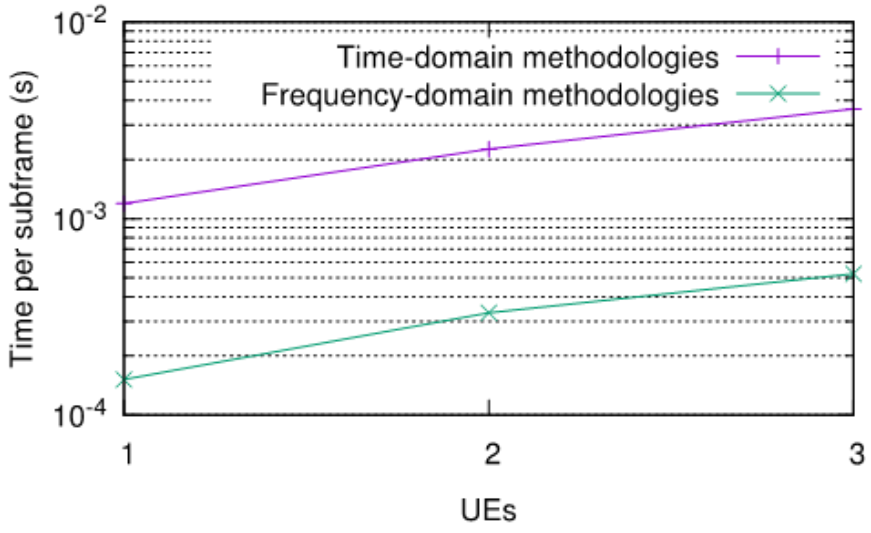
Table 1. Network emulation parameters

Emulation parameters	Value
Band	7
Transmitter power	15 dBm
Working mode	FDD
Cyclic prefix	Normal
Interface compression	A-law
System bandwidth	5 MHz
Transmission mode	SISO (1)
Multipath channel model	AWGN
Interface	IF4P5

Source: Own source.

In figure 10, we report the frequency-domain methodologies have the average computation time 10-fold faster than time-domain methodologies [17]. This fact is a physical layer achievement that is independent of scheduler tasks and allows real-time emulations (we define real-time emulations where the average computation time of the multipath channel takes less than 1ms or 1 subframe).

Figure 10. Comparison of the average computation time for frequency-domain and time-domain methodologies using the AWGN channel



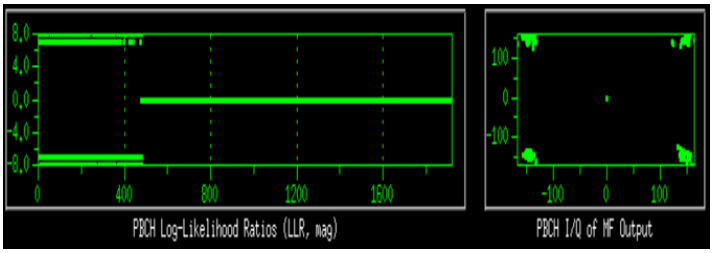
Source: Own source.

Allocation of Physical Resource Blocks

We modified the *dlsch_modulation* () function to visualize PRBs at the RCC. By default, the OAI platform sends a small amount of downlink traffic to the UE (around 15Kb/s) to maintain it in RRC connected state. Only the four central PRBs in subframes 0 and 5 are used for common channels (BCCH, CCCH, PCCH, and MCCH), and they are skipped from the PRBs allocation.

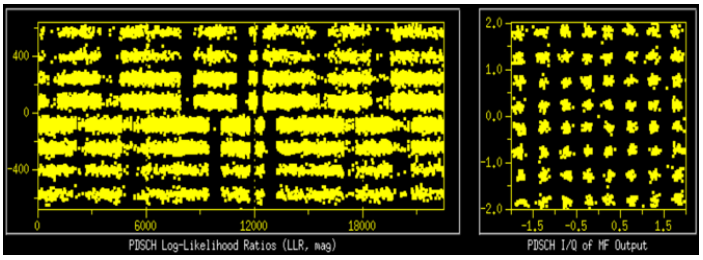
We sent UDP downlink traffic from the EPC to UEs by using the network performance measurement tool called iperf [20]. We visualize the PBCH and PDSCH with downlink UDP traffic in figures 11 and 12, considering the Additive White Gaussian Noise (AWGN) channel. This channel model is recommended for frequency-domain methodologies because the inter-symbol interference is not avoided, other channel models will increase the block error rate compared to time-domain methodologies. Further improvements for the turbo encoding and interleaving are required, but they need a higher computational complexity. Nevertheless, this issue also remains as future work.

Figure 11. PBCH Log-Likelihood radio using frequency-domain methodologies (AWGN channel)



Source: Taken from the OpenAirInterface xforms softscope

Figure 12. PDSCH Log-Likelihood radio using frequency-domain methodologies (AWGN channel)



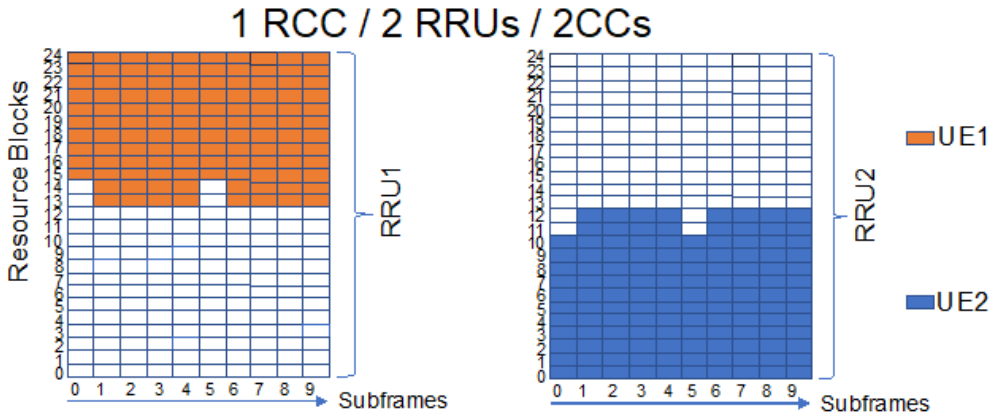
Source: Taken from the OpenAirInterface xforms softscope

We considered a software-only scenario that employs one EPC, one RCC (two CCs, one with an active UE), and two RRUs. We validate the allocation of PRBs using the CS technique and frequency-domain methodologies as outlined in figure 13. In that case, the UE2 allocates roughly the first half of the spectrum (13 PRBs) and UE1 the remaining (12 PRBs). For a

better understanding, we show in figure 14, the allocation when the emulator lte-softmodem (RCC side) is executed. The UE1 connects the network through RRU1, and the UE0 links the network through RRU0. The USRP B200mini-i and the Samsung Galaxy S8 were employed to validate the allocated PRBs. It was achieved doing a restriction by software for the first and second half of total PRBs. The result was the same allocation of figure 13, but only for the scenario of one RRU and one UE.

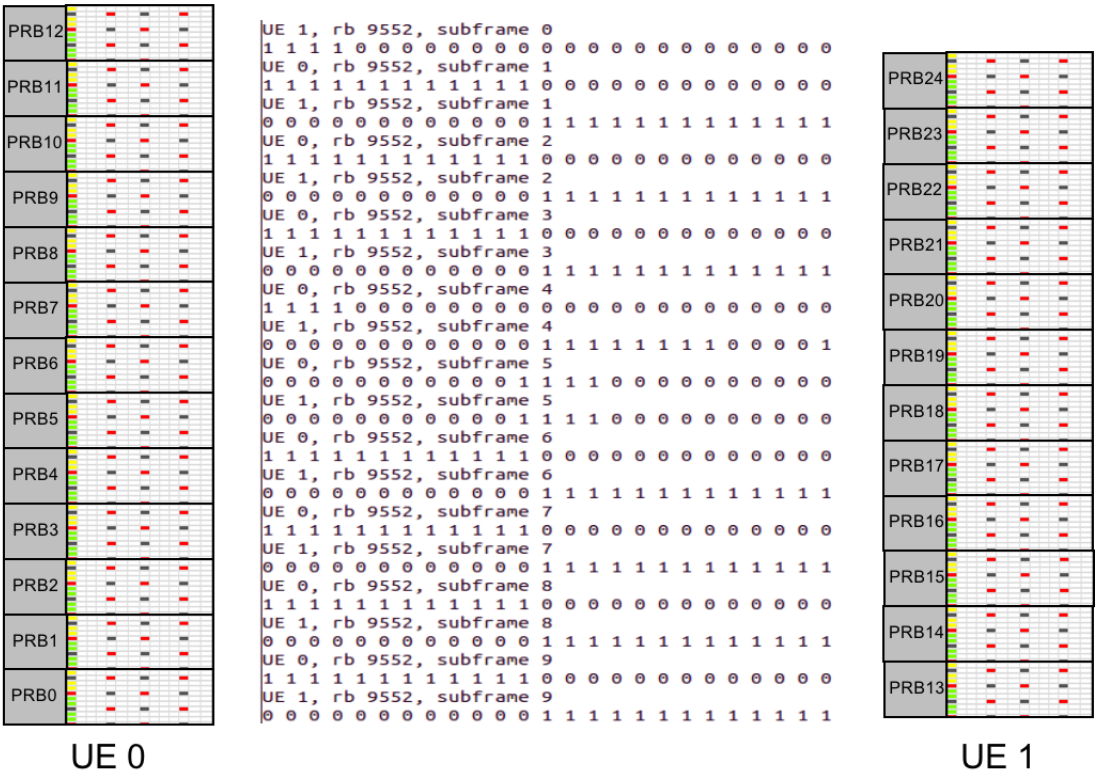
It is essential to point out that the software-only emulation brings us a new framework to analyze system-level, real-time, and 3GPP standard-compliant scenarios in OAI, which is very useful at the first stage of prototyping applications for C-RANs. In this paper, we considered the static coordinated scheduling, but other techniques such as carrier aggregation can be easily extended.

Figure 13. Allocation of resource blocks employing the static load-unaware coordinated scheduling (AWGN channel). Roughly half of the resource blocks are associated with each UE



Source: Own source.

Figure 14. Allocation of resource blocks employing the static load-unaware coordinated scheduling (AWGN channel). Roughly half of the resource blocks are associated with each UE



Source: Own source.

Average Computation Time

Some profiling programs collect time statistics such as Gprof, Valgrind, and Perf, but they are not susceptible to real-time because they slow down OAI emulations. We benchmarked previous profiling programs and the OpenAir performance profiler to measure the average computation time. The last one was employed because is the fastest, the most efficient, and less time-consuming.

Afterwards, we defined two scenarios: the first one with two RRUs and two UEs, and the second one with three RRUs and three UEs in a software-only environment. Each active UE is attached to the network through different CCs, and this task is accomplished at random. In table 2, we present the average computation time employing time-domain and frequency-domain methodologies. It demonstrates that frequency-domain methodologies allow the execution of real-time and scalable emulations. However, emulations using time-domain methodologies have synchronization issues that make them very difficult to achieve (This is

the reason why the scenario with three RRUs using time-domain methodologies does not have a value of the average computation time).

Table 2. Average computation times of the multipath channel

RCCs	CCs RRUs	UEs	Time-domain methodologies in μs	Frequency-domain methodologies in μs
1	2	2	2215,87	335,2
1	3	3	N/A	455,23
1	4	4	Non-real-time zone	

Source: Own source.

Conclusions

We successfully emulated real-time static load-unaware coordinated scheduling prototype for C-RANs in a software-only environment using frequency-domain methodologies and taking advantage of the IF4P5 interface. This accomplishment is the preamble for 3GPP standard-compliant validation of applications in large-scale scenarios where the bottleneck of the multipath channel's computation time is not a problem. The validation of the CS with the USRP B200 mini-i accomplished the allocation of PRBs, but it was restricted to one RRU and one UE.

Our proposal is a framework to test real-time and system-level emulations in a software-only environment. It evaluates the reproducibility and scalability of prototyped techniques. Also, it does not require radio devices such as software-defined radio frontends, reducing prototyping uncertainties of the physical hardware. In any case, the hardware can be connected to create a hybrid emulation without problems (for example, a real Nokia EPC can be easily changed for the OAI EPC based on software).

Future work will include dynamic coordinated scheduling and inter-cell interference computation in a hybrid scenario composed of synthetic and real networks. It will increase OAI's capability to manage large-scale network emulations, as it is one of the 5G strategic areas of the OpenAirInterface Software Alliance.

References

[1] F. Kaltenberger, G. de Souza, R. Knopp, and H. Wang, "The OpenAirInterface 5G New Radio Implementation: Current Status and Roadmap," *WSA 2019, 23rd Int. ITG Work. Smart Antennas*, pp. 134–138, 2019. Available: https://5genesis.eu/wp-content/uploads/2019/04/The-OpenAirInterface-5G-New-Radio-Implementation_Current-status-and-roadmap.pdf

[2] J. Huang, Y. Yuan, and S. Ma, "5G Mobile Communications,"Springer, pp. 431–455, 2017. https://doi.org/10.1007/978-3-319-34208-5_16

[3] I. Chih-Lin, J. Huang, R. Duan, C. Cui, J. Jiang, and L. Li, "Recent progress on C-RAN centralization and cloudification," *IEEE Access*, vol. 2, pp. 1030–1039, 2014. doi:10.1109/ACCESS.2014.2351411

[4] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent advances in cloud radio access networks: System architectures, key techniques, and open issues," *IEEE Commun. Surv. Tutorials*, vol. 18, no. 3, pp. 2282–2308, 2016. Available: https://arxiv.org/pdf/1604.00607.pdf

[5] J. Acharya, L. Gao, and S. Gaur, *Heterogeneous networks in LTE-Advanced*. West Sussex: Wiley, 2014.

[6] M. Peng, T. Q. S. Quek, G. Mao, Z. Ding, R. Knopp, and C. Wang, "Artificial-intelligence-driven fog radio access networks: Recent advances and future trends," *IEEE Wirel. Commun.*, vol. 27, no. 2, pp. 12–13, 2020.

[7] N. Nikaein *et al.*, "OpenAirInterface: A flexible platform for 5G research," vol. 44, no. 5, pp. 33–38, 2014. Available: http://www.sigcomm.org/sites/default/files/ccr/papers/2014/October/0000000-0000004.pdf

[8] OpenAirInterface, "OpenAirInterface: 5G software alliance for democratising wireless innovation," 2020. Available: https://openairinterface.org

[9] N. Iardella *et al.*, "Coordinated scheduling in a Virtual-RAN prototype with OpenAirInterface," June, 2016. Available: http://www.iet.unipi.it/g.stea/papers/EuCNC_2016.pdf

[10] N. Iardella *et al.*, "Flexible dynamic coordinated scheduling in virtual-RAN deployments," *2017 IEEE Int. Conf. Commun. Work. ICC Work. 2017*, no. 2, pp. 126–131. Available: http://pages.di.unipi.it/frangio/papers/IEEEICC2017.pdf

[11] A. M. Alba, A. Basta, J. H. G. Velásquez, and W. Kellerer, "A realistic coordinated scheduling scheme for the next-generation RAN," in *2018 IEEE Global Comm. Conf., GLOBECOM 2018, Proc*.

[12] Fujitsu, "Activity of FUJITSU in OAI," 2017. Available: https://www.openairinterface.org/docs/workshop/3_OAI_Workshop_20170427/plenary/Yuko_Akiyama_-_Activity_of_FUJITSU_in_OAI.pdf

[13] Mosaic5G, Leveraging an Ecosystem of 5G Services, 2019 [Online]. Available: http://mosaic-5g.io/resources/mosaic5g_flexran.pdf. Accessed on: September 22, 2019].

[14] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A survey of the functional splits proposed for 5G mobile crosshaul networks," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 146–172, 2019. doi: 10.1109/COMST.2018.2868805

[15] C. Cox, *An introduction to LTE. LTE, LTE-Advanced, SAE and 4G Mobile Communications*, Second ed. Wiley, 2014.

[16] A. Virdis, N. Iardella, G. Stea, and D. Sabella, "Performance analysis of OpenAirInterface system emulation," *Proc. 2015 Int. Conf. Futur. Internet Things Cloud, FiCloud 2015, Int. Conf. Open Big Data, OBD 2015*, pp. 662–669.

[17] L. Ariza, R. Knopp, and J. Eslava, "Real-time emulation methodologies for centralized radio access networks," *2019 IEEE 20th Int. Work. Signal Process. Adv. Wirel. Commun.*, pp. 1–5, 2019.

[18] L. Ariza, "Master_large_scale_emulations repository," 2019 [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/tree/master_large_scale_emulations. Accessed on: September 22, 2019.

[19] R. Knopp, "Some preliminaries," 2020 [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/-/blob/develop/targets/DOCS/oai_L1_L2_procedures.pdf. Accessed on: June 23, 2020.

[20] J. Dugan, E. Seth, A. M. Bruce, J. Poskanzer, and P. Kaustubh, "iPerf: The ultimate speed test tool for TCP, UDP, and SCTP," 2019. Available: https://iperf.fr/.