MARC: On Modeling and Analysis of Software-Defined Radio Access Network Controllers

Arled Papa*, Raphael Durner*, Endri Goshi*, Leonardo Goratti†, Tinku Rasheed†, Andreas Blenk*, Wolfgang Kellerer*
*Chair of Communication Networks, Technical University of Munich
†Zodiac Inflight Innovations, Weßling Germany
*arled.papa@tum.de, *r.durner@tum.de, *endri.goshi@tum.de, †leonardo.goratti@zii.aero, †tinku.rasheed@zii.aero, *andreas.blenk@tum.de, *wolfgang.kellerer@tum.de

***Abstract*—Network programmability also sneaked into the mobile world leading to the emergence of Software-Defined Radio Access Network (SD-RAN) architectures. Interestingly, while only a small number of prototype architectures exist for SD-RAN, their performance evaluations are unfortunately also limited. Recent evaluations are carried out for small network dimensions of up to 50 devices, while emerging 5G/6G networks envision numbers of devices beyond 5000. Although 5G/6G applications are more stringent with respect to latency guarantees, performance evaluations of such low scale remain questionable. To fill this void, this paper presents `MARC`: a novel benchmarking tool for SD-RAN architectures and their controllers. We use `MARC` to measure, analyze and identify performance implications for two state-of-the-art open source SD-RAN solutions: `FlexRAN` and `5G-EmPOWER`. We perceive results for monitoring application scenarios considering fully centralized control. For this setting, our findings show that the proposed architectures with a single SD-RAN controller are not scalable and can even lead to unpredictable network operations. Using our tool and based on our insights, we provide and implement design guidelines for the internal working behavior of the existing controllers.**

***Keywords*—SD-RAN, Benchmarking, SD-RAN Controllers, Performance Management, Network Measurements.**

I. Introduction

Next generation 5G/6G wireless networks need to cope with applications such as tele-operations, online gaming and Internet of Things (IoT) with thousands of connected devices [1]. Current networks cannot support these wide range of heterogeneous and delay stringent applications, thus research as well as 3GPP standardization [2] propose new emerging techniques to cater for such diverse requirements.

In order to support heterogeneous applications, emerging solutions suggest a full softwarization of the mobile network by means of Software-Defined Networking (SDN). On the one hand, SDN alleviates the deployment of network slicing [3], which leads to the ability to deploy and maintain large networks with thousands of devices. On the other hand, SDN is also intended to boost the network performance by flexibly deploying functionalities closer to the users such as Mobile Edge Computing (MEC), to cater for the stringent delay requirements of 5G/6G applications.

Recently, the concept of network programmability and virtualization has also been embraced by the industry. As reported by Rakuten, Inc. [4], the programmability provided by SDN can lead to reliable network deployments. Additionally, in the context of next-generation Radio Access Networks (RANs), the O-RAN alliance is investing in open, intelligent and virtualized systems [5], demonstrating the importance of softwarization in the 5G/6G era.

However, while softwarization has been mainly applied in data centers and wired networks [6], [7], only recently the attention has been shifted towards the network edge. In particular, RAN can benefit from the centralization concept of SDN in order to improve spectrum efficiency, power management, interference cancellation and handovers [8]–[10], which require tight cooperation among Base Stations (BSs). Hence, the SDN paradigm has been adopted by the RAN as lately shown in standardization [11], where a clear separation of the control and data plane is proposed. In the heart of the SD-RAN architecture lies the SD-RAN controller, where the control over the whole network is centralized. Intuitively, the SD-RAN controller becomes a potential bottleneck given the high signaling overhead imposed by thousands of devices. While vast ongoing research is based on concepts containing SD-RAN controllers [12]–[21], unfortunately only a few performance evaluations are conducted [8], [9] for small network dimensions up to 50 devices.

The main research question we address is whether current SD-RAN controllers can sustain the scale of future networks [1], such as IoT (e.g., > 5000 connected devices). We identify the lack of a scalable performance measurement tool as a problem since the performance guarantees in a realistic deployment remain unknown and the possibility for commercial use is questionable. Nonetheless, designing a tool that is able to scale to realistic 5G/6G network dimensions is non-trivial. On the one hand, current open source SD-RAN controllers are implementation specific: they are designed for OpenAirInterface [22] or srsLTE [23] 5G platforms. That means that the interoperability among platforms is not attainable. This hinders the combination of the best features from each platform. Furthermore, OpenAirInterface [22] and srsLTE [23] can only be operated utilizing expensive wireless

hardware [24], which makes the extension of testing platforms infeasible. Hence, scaling the network would require a tremendous investment in space and time for research and testing purposes. On the other hand, the SD-RAN paradigm is still not standardized. Therefore, the extensibility towards new SD-RAN controllers without changing the system itself requires an intensive study of the design and implementation of SD-RAN benchmarks.

To overcome this issue, in this paper we present MARC, a novel benchmarking tool for SD-RAN architectures and their controllers. We use MARC to measure and analyze two state-of-the-art open source solutions: FlexRAN [8] and 5G-EmPOWER [9]. MARC extends the controllers' performance analysis of previous works to larger network dimensions (e.g., 5000 network components). By considering the scenario of monitoring applications under fully centralized control, we provide design guidelines to overcome performance bottlenecks for FlexRAN with respect to memory resource allocation. Finally, we show that current SD-RAN architectures do not scale, thus we conclude that alternative approaches are necessary to increase the scalability. The detailed contributions of our work are summarized as follows:

- We perform a state-of-the-art analysis on SD-RAN based architectures and existing benchmarking tools in Section II, while identifying the limitations of existing performance evaluation approaches in terms of scalability in Section III. For this, we deeply analyze the code of the available open source SD-RAN controllers.
- We design a data plane performance behavior model for SD-RAN controllers to generate user activation and deactivation events based on traffic patterns defined by the standardization in Section IV-B4.
- We implement a benchmarking tool based on the design of a novel SD-RAN control plane model and make use of the developed data plane model to test the scalability of open source SD-RAN controllers in Section IV-B3.
- With the help of MARC we demonstrate the limitations of SD-RAN controllers considering monitoring applications under fully centralized control and provide an insightful guideline for overcoming performance bottlenecks in Section V. Moreover, we show that current SD-RAN architectures with a single SD-RAN controller do not scale, thus we discuss a potential alternative architecture to increase the scalability in Section VI.
- We publish the code of our benchmark [25] to further investigate current and emerging SD-RAN controllers.

We start our paper with detailing the related work for benchmarking on the core network and RAN in Section II. Furthermore, we provide a description of existing open source SD-RAN controllers in Section III and demystify their implementation principles and protocols. In Section IV we introduce our benchmarking tool and detail the measurement setup and output metrics for our analysis, whereas we report on results for our measurements in Section V. After we discuss the main findings of our work in Section VI, we highlight the benefits of our benchmark and conclude our paper in Section VII.

II. Related Work

The concept of Software-Defined Networking (SDN) has recently triggered vast amount of ongoing research in the Radio Access Network (RAN). From an architectural perspective, several conceptual works [26]–[31] address the potentials of SD-RAN and propose techniques to overcome vital RAN challenges such as resource management, co-ordination, handover and interference cancellation. However, they lack practical implementations.

Recently, softwarized prototypes have emerged in the state-of-the-art to fill this void [8], [9], [32]. FlexRAN [8] is the first open source SD-RAN controller platform that provides a separation of the control and data plane in the RAN and enables a centralized control functionality. FlexRAN is built for OpenAirInterface [22], which in turn is a software platform that allows for 4G/5G experimentation. Similar to FlexRAN, 5G-EmPOWER [9] is another experimental platform built on top of srsLTE [23] and provides a control-data plane separation for RAN. Finally, Orion [32] is an extension of FlexRAN and provides an approach for efficient RAN slicing. However, unlike the previous mentioned platforms, Orion is not open source and therefore its potentials cannot be fully exploited by researchers in the field.

The above experimental platforms equip academia with powerful tools and provide the possibility to contribute with practical experiments. This enables faster progress towards satisfying the new 5G requirements [2], and at the same time triggers a closer collaboration between industry and academia. However, while the current evaluation provided for 5G-EmPOWER [9], Orion [32], OpenAirInterface, srsLTE in [33] and FlexRAN [8], [34] is sufficient for academic purposes, the possibility for a commercial use requires further investigation with focus on performance guarantees in realistic network dimensions, which is currently limited.

In [33], an evaluation is conducted for OpenAirInterface and srsLTE, highlighting a limited number of 4 User Equipments (UEs) on 4G/5G softwarized prototypes [20], [22], [23], mainly due to design choices or to avoid the purchase of expensive hardware for each UE. Regarding SD-RAN controllers, Foukas et al. [8] provide an analysis of CPU and memory utilization as well as signaling overhead for FlexRAN. However, only 4 Base Stations (BSs) and 16 emulated UEs are considered and thus the real scalability of the controller remains unknown. In [34] an evaluation is provided for the CPU and memory utilization of FlexRAN with up to 250 emulated BSs. Nonetheless, no UEs are considered, thus an important source of signaling overhead is neglected. Recently, a new FlexRAN version has been introduced by Mosaic5G [35], which improves the RAN slicing capabilities and provides handovers. However, to the best of our knowledge, deep performance evaluations have not been performed yet compared to the original FlexRAN [8]. Furthermore, Conorado et al. [9] evaluate 5G-EmPOWER with respect to the CPU utilization with varying number of emulating BSs and UEs for a handover use case. However, implementation details regarding the emulation of the BSs and UEs as well as a signaling overhead analysis are not provided.

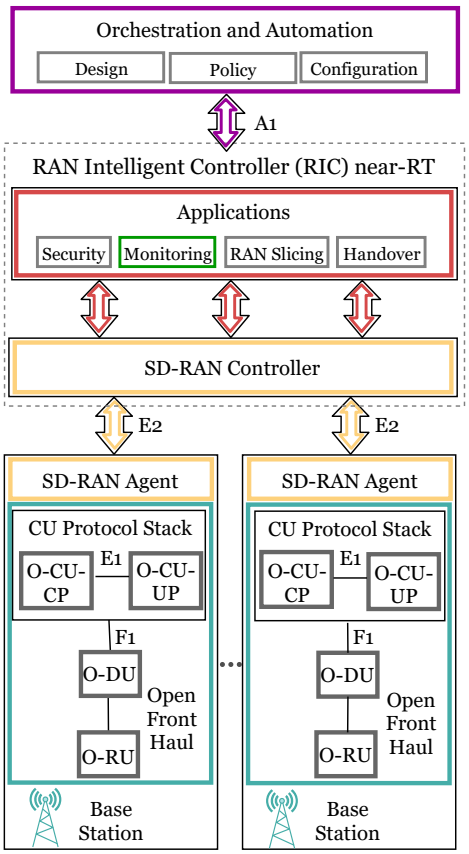


Figure 1: High level overview of the SD-RAN paradigm according to the O-RAN terminology [5]. An SD-RAN agent resides at each Base Station in order to enable the communication with the SD-RAN controller. The SD-RAN controller provides an abstraction overview of the underlying network to the applications and enables them to modify Base Station's parameters such as power, scheduling, modulation and coding. This block is referred to as **near real time controller (near-RT)**. The near-RT controller is then connected with the orchestration and automation layer, responsible for design and network policy configuration.

To fill the void, we design MARC an SD-RAN benchmarking tool able to provide a detailed performance evaluation for FlexRAN [8] and 5G-EmPOWER [9] considering up to 5000 connected devices. The implementation of MARC is based on similar benchmarking tools for SDN controllers and Hypervisors [36]–[39], which have proven to be successful and beneficial for the fixed network side, yet not available for RAN. MARC distinguishes from existing SDN controller tools, since it includes the protocols of SD-RAN and incorporates specific signaling information unique for RAN. That said, additional mobile protocol stack details have to be implemented.

III. Background

In this section we introduce the concept of Software-Defined Radio Access Networks (SD-RANs). We focus on two open source SD-RAN controllers in the literature: FlexRAN [8] and 5G-EmPOWER [9] while detailing their main functionalities and protocols. Due to lack of information regarding the protocol designs of FlexRAN and 5G-EmPOWER in the original works, we provide more details as it is crucial to the design of our benchmarking tool, as well as it already hints potential performance bottlenecks for the SD-RAN platforms. That is, we identify the protocol behavior by deeply analyzing the software code.

A. Software-Defined Radio Access Networks

The main concept of SD-RAN lies on the control functionality over the RAN. This implies a full separation of the data and control plane of the RAN, where the control resides in devices known as SD-RAN controllers. In turn the underlying data plane components (i.e., Base Stations (BSs)) communicate with SD-RAN controllers by utilizing control protocols similar to OpenFlow [40], which is one of the well known SDN protocols in the core network.

A high level overview of the SD-RAN paradigm is given in Fig. 1. As presented, SD-RAN controllers abstract the underlying network for the applications. The nature of such applications can vary from monitoring to operational and management. To facilitate the communication between the SD-RAN controllers and the underlying network, BSs are equipped with control components referred to as SD-RAN agents. Moreover, protocol functions within BSs are structured according to the proposed O-RAN splits [5], namely, Central Unit Control Plane (CU-CP), Central Unit User Plane (CU-UP), Distributed Unit (DU) and Radio Unit (RU) as demonstrated in Fig. 1. However, for the sake of simplicity, for the remainder of the paper we will refer to a BS as a single component.

SD-RAN agents understand the SD-RAN protocol and are responsible for translating and enforcing the SD-RAN controller's decisions on the underlying BS devices. While SD-RAN agents reside at the BSs, SD-RAN controllers can be deployed on edge servers and communicate via TCP with the SD-RAN agents. Finally, User Equipments (UEs) are served by BSs in order to enable communication. The block composed of the SD-RAN controller and applications, is referred to in the O-RAN terminology [5] as **near real time RAN intelligent controller (near-RT RIC)**. Furthermore, the near-RT RIC is connected with the orchestration and automation layer in order to provide the design, policy and configuration of the network. In that regard, both FlexRAN and 5G-EmPOWER can be grasped as near-RT RICs according to O-RAN.

In RAN the main challenges include an efficient allocation of the scarce radio resources, but also management of handovers and interference. To achieve these goals, efficient scheduling, interference cancellation and handover algorithms are built in the SD-RAN controllers [8]–[10]. A detailed overview of the main features supported by FlexRAN and 5G-EmPOWER are summarized in Table I.

B. FlexRAN

FlexRAN [8] is an SD-RAN platform built on top of OpenAirInterface [22], which in turn is an open-source research platform for 4G/5G communication. In this subsection

Features	FlexRAN	5G-EmPOWER
Base Station reports	✓	✓
UE reports	✓	✓
Trigger events	✓	✓
Control delegation	✓	✗
Real time controller	✓	✗
Security	✗	✓
Handover	✗	✓
RAN slicing	✓	✓

Table I: Overview of the features supported by FlexRAN and 5G-EmPOWER SD-RAN controller platforms.

we initially explain the FlexRAN architecture and later the protocol used for communication between the FlexRAN controller and FlexRAN agents.

1) Architecture: The FlexRAN architecture, follows the overall SD-RAN concept presented in Fig. 1. The summary of FlexRAN's features is portrayed in Table I. FlexRAN provides an abstraction layer for applications to be able to monitor, change or adapt several parameters of the network (i.e., power, resource allocation, modulation and coding schemes), based on request, utilizing REST Application Programmable Interfaces (APIs). The platform consists of the master FlexRAN controller, and the FlexRAN agents. The FlexRAN agents, are responsible for enforcing all the control rules with respect to resource allocation, handovers on the underlying physical network. A single dedicated FlexRAN controller is envisioned for the control over multiple agents, each one representing a physical BS. However, FlexRAN can delegate the control to the local FlexRAN agents at runtime to reduce the delay. The communication among them is realized by using the FlexRAN protocol.

2) Protocol: The FlexRAN protocol defines the set of rules used for the communication on the control plane among the master FlexRAN controller and the FlexRAN agents. The full flow diagram of the protocol is presented in Fig. 2. Due to lack of detailed elaboration in the original work [8], an extensive analysis was performed on the platform to be able to derive the exact protocol. We have divided the protocol diagram into 5 phases, each of them representing one functionality or event. Initially, a **hello handshake** message is needed to establish the connection and registration of the agent to FlexRAN controller's database. Further, the FlexRAN controller performs a round of requests regarding BS configurations, User Equipment (UE) configurations and Logical Channel (LC) configurations. Consequently, the agent replies with all the necessary information to the FlexRAN controller terminating the **configuration phase**. Once the initial setup has terminated, the controller requests a **periodic statistics report** analysis from the agent. The report includes information with respect to the BS and active UEs. Such reports are periodically transmitted with a frequency of 5ms to provide a real time network monitoring.

Additionally, the FlexRAN protocol does not only support periodic events, but it is also triggered by events such as UE activations/deactivations in the underlying data plane. In case of an activation event, the agent notifies the FlexRAN controller with a UE change state (i.e., **UE activation**).

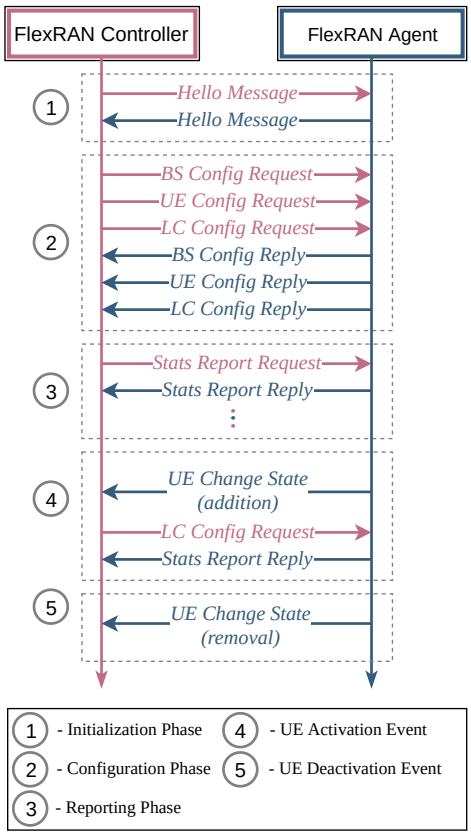


Figure 2: Overview of the FlexRAN protocol. The protocol is split into 5 phases each one representing a set of messages exchanged between the FlexRAN SD-RAN controller and the FlexRAN agent.

Immediately, the FlexRAN controller requests information regarding the UE (i.e., LC configuration request). In turn the agent provides them to the controller and inserts the newly joined UE in the list of active UEs. This list is used to fill in the information for the next periodic report message. Finally, a trigger message can occur also for a UE change state to **UE deactivation**. The FlexRAN controller keeps all the information acquired from its agents to a database namely RAN Information Base (RIB). RIB is loaded in the memory in order to achieve faster performance. It is updated every $\sim$ 1ms and it is structured as a forest graph, where each agent is a root of the tree and UEs are the leaves. However, storing such frequent updates in the memory can pose challenges when the network dimensions increase. To support real time operation, FlexRAN is designed such that it decouples the writing and reading of the RIB from the applications running on top. This is achieved by assigning 20% of the processing time to a **RIB updater** and 80% to a **task manager** that handles the applications in a non blocking mode.

C. 5G-EmPOWER

Similar to FlexRAN, 5G-EmPOWER [9] is an open source SD-RAN controller platform existing in the state-of-the-art. It similarly follows the general architecture of the SD-RAN paradigm introduced in Fig. 1.

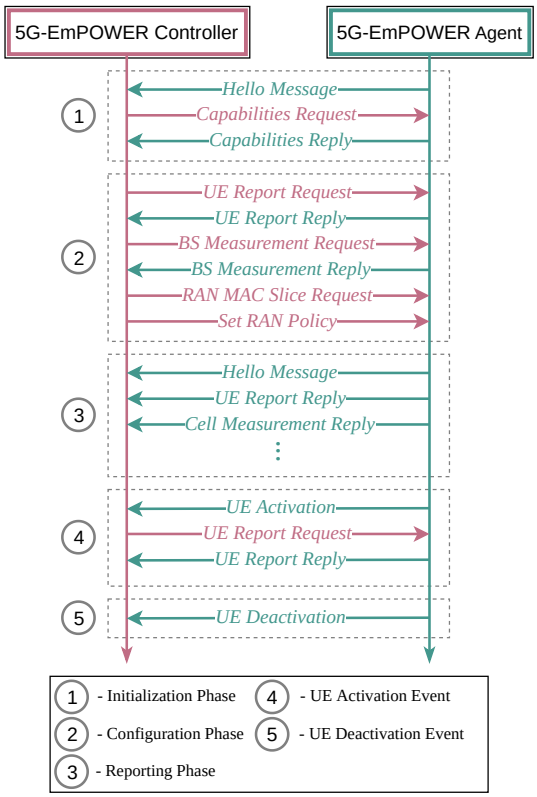


Figure 3: Overview of the OpenEmpower protocol. The protocol is split into 5 phases each one representing a set of messages exchanged between the 5G-EmPOWER SD-RAN controller and the 5G-EmPOWER agent.

1) Architecture: The 5G-EmPOWER platform provides an abstract view of the network including UEs and BSs to various applications running on top and as such enables the modification of scheduling, power management, handover decisions. The communication between the applications and the 5G-EmPOWER controller is realized by means of REST APIs. According to Table I, for 5G-EmPOWER such applications include monitoring, security, handover, and RAN slicing. An underlying physical infrastructure consists of multiple BSs, where each BS is represented by a 5G-EmPOWER agent. In turn all the information from the 5G-EmPOWER agents are gathered in the master 5G-EmPOWER controller.

Differently from FlexRAN, 5G-EmPOWER is not a real time controller i.e., does not adapt to network changes on short time scales like $\sim$ 1ms, instead such adaptations are influenced by instructions sent by the applications. Furthermore, the control of the network cannot be delegated to the local 5G-EmPOWER agents in runtime unlike FlexRAN. These design choices can influence the data plane performance since infrequent updates might not reflect changes in UEs' behavior, which can result in interference or throughput decrease. However, in contrast to FlexRAN [8], 5G-EmPOWER is the only SD-RAN controller that supports security and handover. Finally, the communication in the control plane, among the 5G-EmPOWER controller and the 5G-EmPOWER agents is realized by using the **OpenEmpower** protocol.

2) Protocol: As aforementioned, the communication in the control plane of the 5G-EmPOWER platform is realized by the **OpenEmpower** protocol. The overview of the **OpenEmpower** is provided in Fig. 3. Even for 5G-EmPOWER the protocol has been identified after tracing the actual implementation in combination with information provided in [9]. We have split the protocol into 5 phases. Similar to FlexRAN, an initialization phase consists of a **hello handshake** between the 5G-EmPOWER agent and 5G-EmPOWER controller. Further, the 5G-EmPOWER controller initiates the **configuration phase** by requesting information with respect to the capabilities of the BS, UE reports as well as initiates BS and UE measurements. Once, the controller gathers all the required information, it enforces RAN slicing policies. The third phase consists of a **periodic report** phase which includes hello keep alive messages as well as UE and BS measurement reports every 500ms. While such an infrequent update decreases the signaling overhead of the controller, it does not reflect the latest wireless channel characteristics and it can lead to suboptimal performance.

Similar to FlexRAN, 5G-EmPOWER supports trigger messages regarding the UE activation and deactivation. Once a UE joins the network, the 5G-EmPOWER agent sends a trigger **UE activation** message to the controller, which in turn requests a UE report, similar to the LC configuration message of the FlexRAN controller. Immediately, the agent replies back with a UE report reply. Moreover, it adds the newly joined UE to the list of active UEs, which is needed to construct the report messages. Differently from the FlexRAN agent, the 5G-EmPOWER agent does not concatenate all the UEs into a single report message, but instead sends a separate report message for each UE. While this does not have a great impact with low number of UEs (e.g., $\sim$ 10), yet it can pose challenges when the number of UEs increases to realistic deployments (e.g., $\sim$ 5000). Finally, an agent triggers the controller even in the case of a **UE deactivation**. Even for 5G-EmPOWER a database is implemented to keep statistics for the newly joined agents and UEs. However, in contrast to FlexRAN in order to increase security, 5G-EmPOWER only serves those agents whose ID is already registered in the controller's database. For our analysis, a modification on the original database (i.e., addition of agents' IDs) is necessary to enable the communication of the 5G-EmPOWER controller with our proposed benchmarking tool.

IV. Benchmarking Design

In this section we present MARC, our benchmarking tool for SD-RAN controllers. Initially, we explain the design goals of MARC and further we detail the architecture and implementation choices.

A. Design Goals

The implementation of MARC is designed such that it fulfills goals as described below:

- **Ability to provide statistics:** The benchmarking tool shall be able to provide extensive statistics regarding the performance of the SD-RAN controllers with respect to

CPU and memory consumption, throughput and latency. This feature allows to characterize the controllers and test their performance predictability.

- **Scalability:** MARC should be able to test the performance of SD-RAN controllers considering large number (e.g., $>$ 2000) of connected network components (i.e., BSs, UEs). With this feature, we are able to understand whether current controllers can be used in realistic deployments.
- **Interoperability:** Our benchmarking tool should be compatible with existing open source SD-RAN controllers. Since the SD-RAN protocol is not standardized, each SD-RAN platform is tied to its respective data plane system. That entails that different SD-RAN controllers cannot communicate with other systems. In order to allow interoperability a unified tool is required. MARC offers two SD-RAN protocols as programming libraries that can be used to abstract the data plane for two open source SD-RAN controllers, namely FlexRAN and 5G-EmPOWER.
- **Extensibility:** Finally, MARC should be easily extensible for new controllers. Although SD-RAN protocols differ from each other, they still have a specific structure in common (i.e., initialization, statistics requests/reply, trigger messages). That entails if the new SD-RAN controller follows either one of the already available protocol libraries supported by MARC no changes are required for the operation. Otherwise, an adaptation of the new controller's protocol to fit the common protocol structure is needed. This can be easily achieved by extending the current programming libraries without re-programming the system itself.

B. Architecture & Implementation

As described in III, FlexRAN and 5G-EmPOWER, have different protocols and design characteristics. For instance, the statistics report frequency and concatenation of BS and UE report messages. Thus, in our benchmarking implementation we need to take this information into account and create separate functions for each SD-RAN agent. Our benchmarking tool has $\sim$ 3000 lines of new code. All the functionalities within an agent are implemented in a non blocking manner using asyncio [41], which is a Python-based package that enables a concurrent execution of the processes.

In this subsection we initially present the architecture overview of MARC. Furthermore, we describe the message libraries required for our implementation and detail the control plane model functionalities of our benchmarking design. In order to emulate a realistic environment, apart from the SD-RAN agents representing the BSs, UEs should be included in our model. However, most of the existing prototype platforms built on top of OpenAirInterface [22] and srsLTE [23] are limited to a maximum of 4 UEs [20], [22], [23], [33]. Hence, due to such design limitations and to avoid the purchase of expensive hardware [24], we provide a performance model for SD-RAN data plane traffic based on 3GPP standardization.

1) Architecture: The overview of MARC's architecture is depicted in Fig. 4. Both FlexRAN and 5G-EmPOWER controllers are connected to MARC by means of standard TCP. We

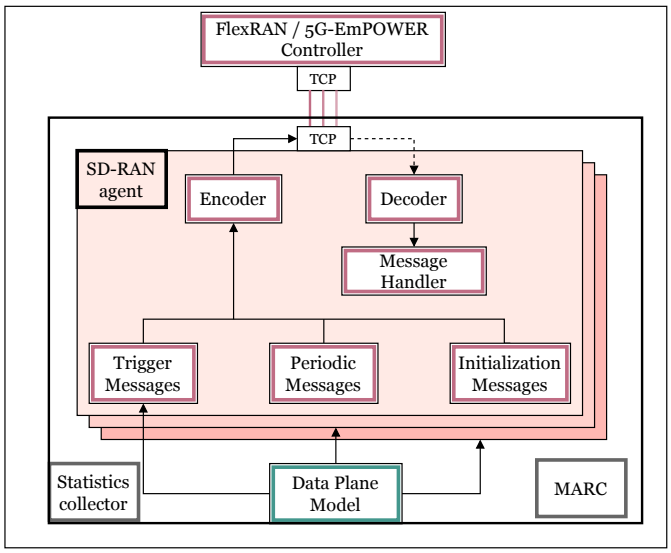


Figure 4: Architecture overview of the benchmarking tool. It portrays the model and implementation of the SD-RAN agents, each one for distinct SD-RAN controllers, FlexRAN and 5G-EmPOWER. The control plane functionalities are depicted with red color, whereas green color depicts the data plane model function. The communication with the SD-RAN controllers is realized by means of standard TCP using *asyncio*, which provides an asynchronous operation.

distinguish between control plane functionalities depicted with red color representing the SD-RAN agents, whereas green color portrays the data plane model function. A statistics collector is implemented in our benchmarking tool in order to provide the gathering and analysis of the measurements. In order to test the scalability of FlexRAN and 5G-EmPOWER our benchmarking tool can generate multiple threads of SD-RAN agents, each with a distinct data plane traffic and TCP port. Moreover, MARC allows for two ways of operation. All the SD-RAN agents can run centralized on a single server, or they can run distributed on different machines. However, in this paper, we demonstrate results for the former approach, where all SD-RAN agents are centralized on a single server.

2) Message Libraries: In order to provide compatibility and interoperability among FlexRAN and 5G-EmPOWER protocols, MARC provides the SD-RAN platform message specific functionalities as programming libraries that can be used to encode/decode and handle messages as follows:

- **Encoding/Decoding**: MARC implements Encoding and Decoding for both **FlexRAN** and **OpenEmpower** protocols as suggested by the original platforms. On the one hand, for the FlexRAN protocol we utilize Google Protocol Buffers [42], which makes use of the mechanism of Protobuf to serialize the structured message types for the communication. On the other hand, for 5G-EmPOWER we utilize the Construct python library [43].
- **Message handler**: The message handler is the core functionality of our benchmarking tool, as it handles all the replies from/towards the controller and behaves according to the protocols. For each SD-RAN agent the

message handler is adapted to act similarly to **FlexRAN** and **OpenEmpower** protocols. Again due to lack of a standardized protocol, the message handler is offered as a library that can be interchanged among the two protocols.

3) Control Plane Modeling: The control plane modeling includes the implementation of functionalities for the agents of FlexRAN and 5G-EmPOWER platforms, whereas the FlexRAN and 5G-EmPOWER controllers, which are the heart of the respective architectures are benchmarked by our tool. Current SD-RAN controllers are tied to OpenAirInterface and srsLTE 4G/5G platforms accordingly. Moreover, the existing operation mode is only available while utilizing expensive wireless hardware, which makes the extension of testing platforms infeasible. In order to overcome this issue, we emulate the FlexRAN and 5G-EmPOWER agents with focus on the control plane functionalities. That entails BS and UE signaling reports, BS initialization messages, BS configuration messages and UE activation and deactivation messages as elaborated in III-B and III-C, respectively. We enhance the operation towards a multi-threading mode, where each agent is accompanied with a single thread to test the controllers' scalability with minimal computing resources. Our main goal is to benchmark SD-RAN controllers considering a high network load. Thus, currently, MARC deploys the extreme case that the network control is delegated fully to the SD-RAN controllers and is not carried out from the BSs. In terms of scalability, this demonstrates a corner case, which can be dynamically changed by offloading control between the SD-RAN controllers and BSs. However, MARC can be extended to enable such a configuration. Initially, that will require the creation of control delegation events on our data plane model depending on 3GPP distributions or realistic traces, following a similar approach to the activation/deactivation events presented later in the manuscript. Moreover, modifications are also required to the message handler to cater for such events in our control plane model. Nonetheless, the effort for including this processing is minimal as it is equivalent to removing BSs and their respective UEs from our measurements dynamically. The main control plane functionalities can be summarized as follows:

- **Trigger message generator:** These type of messages fall into the group of asynchronous messages that are supported both by FlexRAN and 5G-EmPOWER. They mainly consist of new UEs joining or UEs leaving the network. The information received from the data plane model with respect to the UE activation/deactivation is reported to the message handler.
- **Periodic message generator**: Synchronous message type is in charge of generating periodic reports regarding either the UE or BS behavior. In case of FlexRAN such periodicity is more frequent to allow for real time operation $\sim$ 50ms, whereas for 5G-EmPOWER such reports are less frequent $\sim$ 500ms since the real time adaptation is not the focus.
- **Initialization message generator**: Includes all synchronous messages that are exchanged between the SD-RAN controller and its respective agent at the beginning of the connection. As earlier elaborated in Fig. 2 and Fig. 3, respectively, such messages consist of Hello messages, LC configuration, UE and BS configuration.

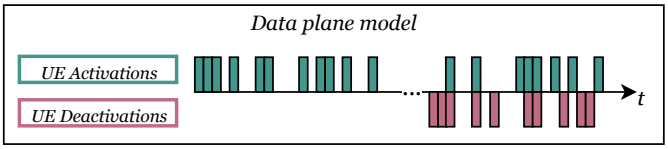


Figure 5: Data plane model utilized for the user traffic generation. It emulates the number of users entering or departing the network, where each event is presented as a tuple consisting of an event *type* and a *timestamp*. The creation of the events is based on an SD-RAN data modeling proposed in this work given standardization parameters.

4) Data Plane Modeling: The data plane modeling is used to emulate UE behavior for each SD-RAN agent. We base our model on 5G specifications [44], where services are separated into enhanced Mobile Broadband (eMBB), massive Machine Type Communication (mMTC) and Ultra Reliable Low Latency (URLLC). The aforementioned services have heterogeneous characteristics including cell area, UE density, service type, inter activation time and duration. Our model is built for any service type, but in our scenario we show results for eMBB, which consists of UEs requiring high throughput.

Once the thread of an SD-RAN agent is initialized, according to our data plane model a series of UE events are generated, representing UE activations and deactivations as portrayed in Fig. 5, where green color represents the UE activation and red color the UE deactivation. Each event is accompanied with a *timestamp*. In our measurements, the UE activations follow an exponential distribution with a mean of 10s per activation, whereas UE deactivations follow a deterministic distribution with a value of 500s. Nonetheless, MARC allows for a reconfiguration of those values according to the scenario. Since extensibility of the benchmarking tool is one of our main design goals, MARC can be connected to any type of UE simulator that generates UE activation and deactivation events in real time in order to test SD-RAN controllers. The only requirement is that events must be accompanied with a type and timestamp.

For each generated UE, resembling a realistic scenario, characteristics with respect to the channel quality, number of requested resources, number of allocated resources, details with respect to the transmission power, modulation and coding scheme are provided. This information is then transmitted to the respective controller as part of the periodic message generator as described in IV-B3. Considering monitoring applications, the information within the periodic messages with respect to the control plane load is not relevant, as the structure of the messages is fixed by the SD-RAN protocol and it only depends on the number of UEs in the network. However, for applications such as scheduling or RAN slicing this information is crucial, as it influences the resource allocation. In that regard, generating realistic UE statistics becomes mandatory. Although in our work we focus on monitoring applications, MARC allows for such a configuration. In the periodic message

construction, each UE's data can be populated by generating statistics from 3GPP distributions or real channel traces and encoded accordingly. This process generates unique information for each UE and can be utilized as input for the SD-RAN controllers, alleviating the extensibility towards more complex, but vital RAN applications such as scheduling and RAN slicing.

V. Performance Evaluation

In this section, we describe the measurement setup and reveal the main findings of our work. Furthermore, we provide useful guidelines to overcome performance bottlenecks for SD-RAN controllers considering monitoring applications under a fully centralized control plane. In that regard, we identify implications with `FlexRAN` when increasing the network dimensions and propose `AltFlexRAN` to fix them. In principle for `AltFlexRAN`, the allocation of the resources among the **RIB updater** and **task manager** of `FlexRAN` is adapted according to the application to avoid read/write memory issues. In the latter, we will provide in detail guidelines for the adaptations of `AltFlexRAN`.

We evaluate the SD-RAN controllers using performance indicators similar to existing SDN controller and Hypervisor benchmarks [36]–[39]. Initially, we measure the controller delay with respect to **SD-RAN initialization**. Moreover, we evaluate the **CPU** and **memory utilization**, where the python library *psutil* [45] is exploited for the analysis. To test the controlling overhead of the SD-RAN controllers, we measure the control message throughput both towards and from the controller (i.e., **transmission & reception rate**) to identify the stable operational region. We show the results both in msg/s and Kbit/s to better highlight the signaling overhead since `5G-EmPOWER` sends more, but smaller messages compared to `FlexRAN`. A monitoring application runs on the SD-RAN controllers. Each measurement is repeated 10 times, each consisting of 10 minutes; the values are recorded every second. In order to exclude the transient phase of the measurements, we remove the 50 initial and last measurement points from our analysis, i.e., 50 seconds. In our measurements, the data plane traffic for the UE activation time is modeled with an exponential distribution with a mean value of 10s per activation. In turn, the UE deactivation time is modeled with a deterministic value of 500s.

Here, we want to stress that our results are similar to the ones reported from the original `FlexRAN` [8] and `5G-EmPOWER` [9] papers regarding a limited amount of UEs and agents, thus demonstrating the effectiveness and credibility of our benchmark. In line with these results, in the remaining section we present the extension towards larger network dimensions as well as additional results for the signaling overhead and SD-RAN agent initialization delay.

A. Measurement Setup

The experimental setup for the measurements consists of 2 Desktop PCs, each one equipped with an Intel(R) Core(TM) i5-3470 CPU, containing 4 physical CPU cores at frequency 3.2 GHz and 8 GB RAM. Both PCs run Ubuntu 18.04.2 LTS with 4.15.0-58-generic kernel. One of the PCs is running the SD-RAN controller of choice, either `FlexRAN` or `5G-EmPOWER` based on the open source found on the original git repositories [46] for `FlexRAN` and [47] for `5G-EmPOWER`. Whereas the other PC operates the modeled SD-RAN agent.

The detailed measurement diagram is portrayed in Fig. 6. `MARC` emulates the BSs and UEs in the network depending on the test parameters of choice. For each emulated BS, a distinct thread and TCP port is created to distinguish among packets sent from alternative BSs. Within a BS thread, UEs are generated according to data from 3GPP standardization [44]. Each UE carries important characteristics such as channel quality, transmission power and modulation and coding information. Moreover, `MARC` utilizes the python library of *asyncio* [41] to guarantee asynchronous operation among the threads.

B. SD-RAN Agent Initialization Delay

Our initial evaluation consists of controller delay measurements regarding the initialization of SD-RAN agents. Such results become particularly interesting for 5G/6G networks with stringent delay requirements, since they demonstrate the ability to sustain next generation applications.

We first initiate m agents, each one serving t UEs. Once all agents have finalized their initialization phase, they only send periodic statistic reports. In order to measure the initialization delay of the newly joined SD-RAN agent in the system, we start the *m+1* agent and record its initialization delay. We conduct 50 experiments while varying the previously initiated SD-RAN agents and UEs generated for each SD-RAN agent. The results are illustrated in Fig. 7. The x-axis represents the number of m initiated agents running t UEs each, whereas the y-axis the recorded delay. As denoted by Fig. 7a, for the `5G-EmPOWER` the initialization delay remains within reasonable values $\sim$ 10ms up to the case when the CPU utilization saturates (i.e., 30 agents, 100 UEs per agent). In such a case we notice a drastic increase of the delay up to 50ms. For the original `FlexRAN` controller the results are presented in Fig. 7b. For `FlexRAN` we observe a similar experienced delay up to 30 agents $\leq$ 10ms. When the number of agents increases further, up to 50, the delay increases drastically $\geq$ 10s. We will identify the sources of those performance changes in detail in the remainder of this section. We repeat the experiment for the proposed `AltFlexRAN` and report the results in Fig. 7c. Based on `MARC`'s guidelines we are able to contain the SD-RAN initialization delay to 1s even for 50 agents and 400 UEs per agent.

C. CPU Utilization

The main findings for the CPU utilization are shown in Fig. 8. The x-axis represents the number of SD-RAN agents, while for each SD-RAN agent various UE configurations are denoted. The y-axis depicts the CPU utilization in percent % and it is averaged for all the 4 physical cores of the setup.

As we can observe from Fig. 8a, the CPU utilization of the `5G-EmPOWER` controller is increasing almost linearly with respect to the number of agents and UEs. Similar to the results

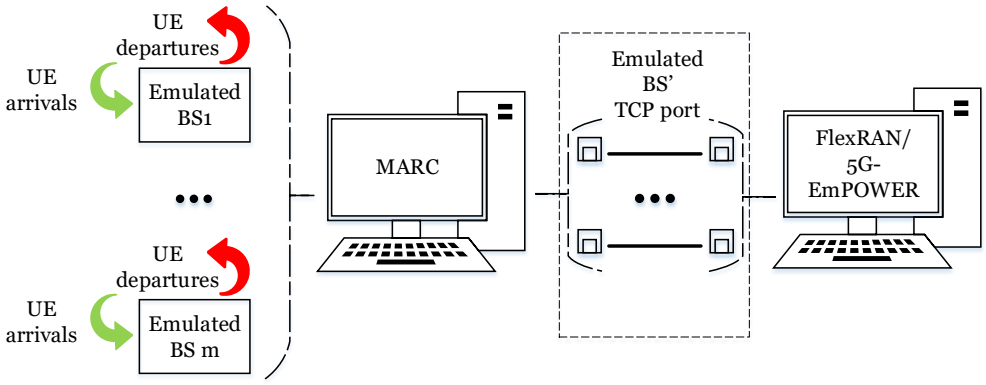
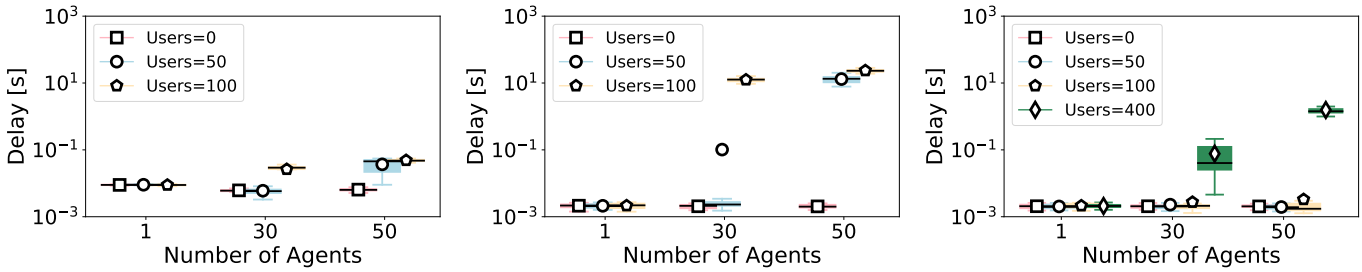


Figure 6: Measurement setup diagram. It portrays the setup for the FlexRAN and 5G-EmPOWER controllers running on separate PCs. Moreover, MARC runs in another PC. MARC emulates the Base Stations as well as the users for the evaluation. Each user is generated according to data acquired from 3GPP standardization and is accompanied with information with respect to channel statistics, transmission power, modulation and coding scheme information, resembling a realistic scenario.



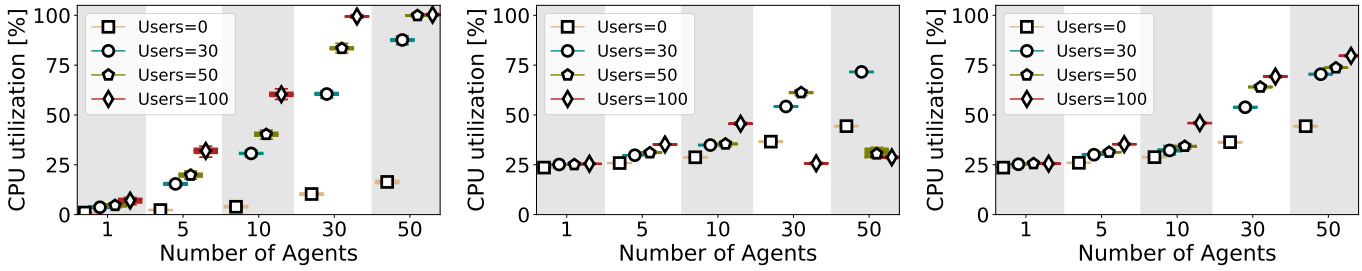
(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 7: Performance evaluation: Initialization delay in seconds. 5G-EmPOWER demonstrates an SD-RAN agent initialization delay of $\sim$ 10ms, whereas the delay for FlexRAN increases up to 10s. AltFlexRAN, however can sustain up to 50 agents with 100 UEs each, experiencing up to 1s delay.

reported in [9] with hardware for 1 SD-RAN agent and 2 UEs, our tool records CPU consumption of $\sim$ 2%. However, the CPU utilization saturates for the configuration of 30 agents and 100 UEs per agent and reaches 100%. We observe an identical behavior also for the configuration of 50 agents and for more than 50 UEs per agent. When no UEs exist in the system (i.e., only BS reports), the CPU load does not increase more than 20% up to 50 considered agents. Nonetheless, once UEs are served, the CPU load increases drastically.

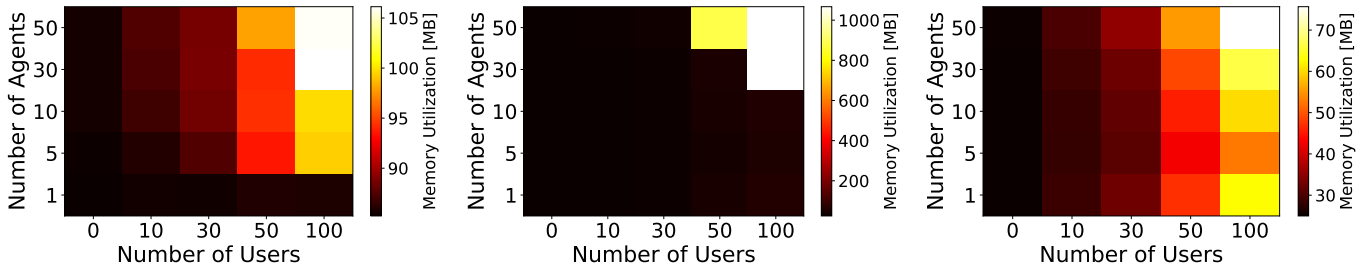
Similar to 5G-EmPOWER the CPU load for varying number of agents and UEs is demonstrated for FlexRAN in Fig. 8b. Again MARC records comparable results with FlexRAN using hardware for 1 SD-RAN agent and 2 UEs of $\sim$ 20%. Even for FlexRAN we notice an increase of the CPU with increasing number of agents and UEs. Differently from the 5G-EmPOWER we notice a CPU drop when reaching a high load (i.e., $\sim$ 30 agents and 100 UEs per agent). Such a drop is also observed for the case of 50 agents when the number of UEs increases further than 50 per agent. Hence, we conclude that FlexRAN cannot support effectively more than 30 agents. For the 5G-EmPOWER controller, the fast CPU increase compared to FlexRAN is clear since for each newly added UE in the system, a distinct UE report has to be sent periodically to the controller. This periodicity corresponds to 500ms. For the FlexRAN controller on the other hand, the UE reports are co-allocated with the BS reports and as such the increasing number of UEs does not affect the number of reports as much. This is notably visible in Fig. 8b, where even when no UEs are present in the system the CPU load is considerably high. It reaches up to 43% when serving 50 agents and no UEs, while a utilization of $\sim$ 22% is observed for the initial scenario with a single agent and no UEs.

Here, we again want to stress that unlike 5G-EmPOWER, FlexRAN is a real time controller. That entails that FlexRAN is designed such that it reflects the latest statistics from the data plane (i.e., wireless channel) in order to adapt the resource allocation (i.e., RAN slicing). Therefore the UE and BS reports are transmitted from the agent to the SD-RAN controller more frequently, every 50ms in our case. Nonetheless, even though the reports for FlexRAN are being sent more frequently, we observe that for the same scenarios, the CPU utilization of FlexRAN unlike the 5G-EmPOWER does not reach 100%. The rational behind this behavior is related to the **RIB updater** function of FlexRAN that was earlier detailed in III-B. The **RIB updater** is responsible for writing and reading FlexRAN agent related information to the controller's database. The original FlexRAN solution suggests an allocation of 20% of the processing time to



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 8: Performance evaluation: CPU utilization [%]. The CPU utilization is increasing almost linearly for all SD-RAN controllers. However, for FlexRAN, the CPU utilization decreases when more than 30 SD-RAN agents are served. AltFlexRAN overcomes this bottleneck by assigning effectively resources between the RIB updater and task manager.



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 9: Performance evaluation: Memory utilization [MB]. 5G-EmPOWER demonstrates a normal memory behavior, while FlexRAN increases the memory consumption with increasing load. We fix this issue in AltFlexRAN by assigning more resources to the RIB updater.

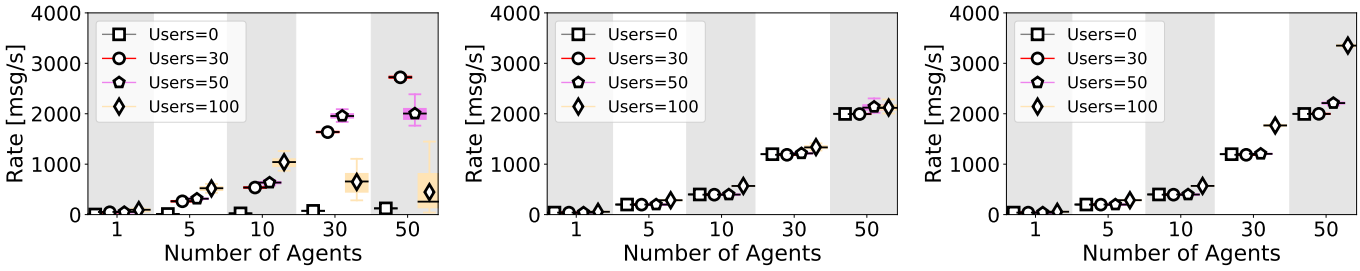
the **RIB updater**, whereas 80% of the processing time is associated to a **task manager** function in charge of handling the applications.

Based on the insights provided by MARC, we adjust the distribution of the resources among the **RIB updater** and **task manager** to adapt to the monitoring application demands i.e., high frequency of statistic reports. Hence, we repeat the experiments for AltFlexRAN by assigning 80% of the time to the **RIB updater** to write/read to/from the RIB and 20% of the time to the **task manager**. As it can be observed in Fig. 8c, the CPU utilization for the AltFlexRAN controller does not drop anymore in a highly loaded scenario, instead it increases. Therefore, AltFlexRAN demonstrates a more stable behavior than 5G-EmPOWER, even though the periodicity of statistics reports is $\sim$ 50ms. Here we can conclude that concatenating the UE and BS reports using the serializing techniques of Google Protocol Buffers [42] consumes less CPU and it is a better design choice for SD-RAN controllers.

D. Memory Utilization

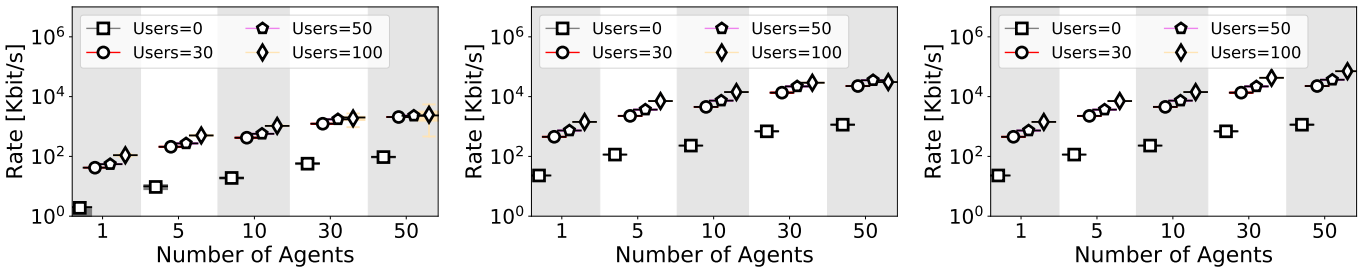
In line with the CPU utilization, we further perform evaluations regarding the memory footprint. Ideally, a well-developed program should not demonstrate drastic memory increase even in a highly loaded scenario. The main results for the SD-RAN controllers regarding the memory utilization are shown in Fig. 9, where the x-axis represents the number of UEs per agent, whereas the y-axis the number of SD-RAN agents. The results are represented in MB. As denoted by Fig. 9a, 5G-EmPOWER experiences a normal behavior of the memory utilization. Although the memory utilization increases with higher number of agents and UEs, it does not exceed 110 MB. Alternatively, FlexRAN demonstrates an unexpected behavior. As illustrated in Fig. 9b for a high number of agents and UEs, the memory utilization keeps increasing by reaching an average of 1 GB for the measurement time (i.e., 10 minutes). However, we stress that if not interrupted the memory can increase up to 100% and therefore even crash the controller.

While at first sight this may indicate a memory leakage, after an intensive analysis of the original FlexRAN code, we identify that the reason behind this behavior is the processing time allocation of the **RIB updater**. In line with the CPU measurements, when a high load is reached, 20% of the processing time initially assigned to the **RIB updater** is not sufficient to sustain a normal behavior. If we recall the original FlexRAN logic elaborated in III-B, the RIB is loaded directly in the memory for faster performance. The increase of the memory indicates that the **RIB updater**, while it can write all the new information retrieved from the agents, it does not have sufficient time to process them. Thus, the memory keeps increasing. To overcome this issue, in AltFlexRAN we switch the processing time of the **task manager** and **RIB updater**. The memory results for the AltFlexRAN are



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 10: Performance evaluation: Reception rate of SD-RAN controllers [msg/s].



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 11: Performance evaluation: Reception rate of SD-RAN controllers [Kbit/s].

presented in Fig. 9c. As we can observe the AltFlexRAN solution is operating in a normal manner and the memory footprint remains within reasonable values $\sim$ 75 MB.

As a conclusion we can say that for FlexRAN a reasonable allocation of the RIB resources depending on the application can overcome potential performance bottlenecks.

E. Transmission & Reception Rate

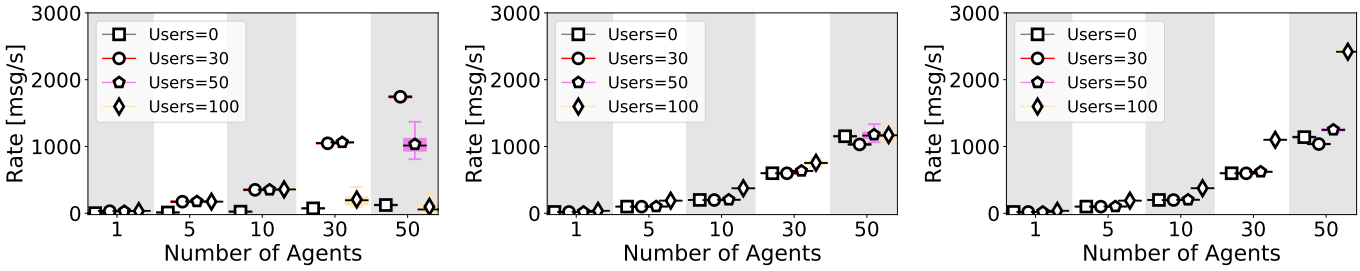
In this subsection we conduct experiments regarding the transmission and reception rate of the SD-RAN controllers and we quantify their signaling overhead. We measure the amount of packets and bytes transmitted and received by the SD-RAN controllers by monitoring the IP tables of the PC where the controller is running. We create two IP table rule chains, one for the reception and one the transmission and capture only the packets destined for our application.

Fig. 10a and Fig. 11a illustrate the number of packets and bytes that the 5G-EmPOWER controller processes with respect to the number of agents and UEs. Although both rates are linearly increasing with increasing number of UEs and agents, at some point (i.e., 30 agents) we notice a decreasing trend, for the cases where the controller is highly utilized, as a result of packet drops and potential TCP effects. We repeat the same experiments for the FlexRAN controller and AltFlexRAN. As we can see from Fig. 10b, Fig. 11b and Fig. 10c, Fig. 11c the controllers' reception rate is increasing with the number of agents and UEs. However, for AltFlexRAN the amount of packets and bytes is higher than FlexRAN: 2500 compared to 1100 msg/s, since we extend the effective operational region. Intuitively, a similar trend is followed by the reception in bytes. Due to the aforementioned design choice differences among FlexRAN and 5G-EmPOWER, the number of packets received by the 5G-EmPOWER in a stable operation region (i.e., up to 30 agents, 50 UEs per agent) is higher compared to FlexRAN: 2100 compared to 1400 msg/s. However, the amount of bytes recorded for FlexRAN is higher compared to 5G-EmPOWER: 70 Mbit/s compared to 1 Mbit/s. This is explained due to the fact that FlexRAN agents concatenate messages together and the size of the packets is higher compared to the 5G-EmPOWER's agents. The signaling overhead recorded for FlexRAN, although higher compared to 5G-EmPOWER, still consumes less CPU resources than 5G-EmPOWER. We again conclude that Google Protocol Buffers [42] are more suitable for SD-RAN.

Compared to controllers' reception rate, the transmission rate is not so demanding since the controllers only reply back to the report statistics with an ACK. For instance for FlexRAN the decrease is almost 2 orders of magnitude: 70 Mbit/s reception compared 1 Mbit/s trasmission rate. A similar trend is observed also for the all the tested controllers. The results of our analysis are portrayed in Fig. 12 and Fig. 13.

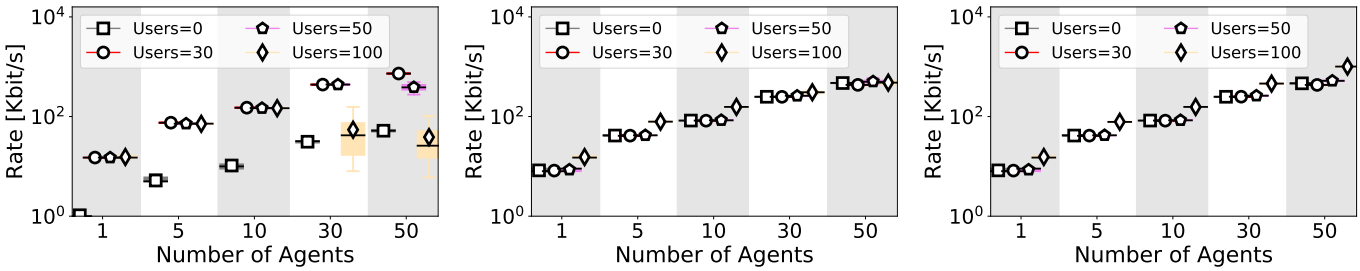
VI. Discussion

While we demonstrated the performance analysis of FlexRAN and 5G-EmPOWER, we discuss whether our benchmarking tool generalizes to other SD-RAN controllers. In principle, MARC is extensible and can be used for any SD-RAN controller that follows the general SD-RAN paradigm explained in III and utilizes one of the considered SD-RAN protocols with no additional changes required. Otherwise, an adaptation of the new controller's protocol to fit the common protocol structure is needed. Nonetheless, this can be achieved



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 12: Performance evaluation: Transmission rate of SD-RAN controllers [msg/s].



(a) 5G-EmPOWER (b) FlexRAN (c) AltFlexRAN

Figure 13: Performance evaluation: Transmission rate of SD-RAN controllers [Kbit/s].

Controllers	Scalability	Signaling Overhead	Initialization Delay
5G-EmPOWER	∼ 30 Agents, 50 UEs	∼ 1 Mbps	∼50ms
FlexRAN	∼ 30 Agents, 50 UEs	∼ 20 Mbps	∼10s
AltFlexRAN	∼ 50 Agents, 100 UEs	∼ 70 Mbps	∼1s

Table II: SD-RAN controllers' observations. A single SD-RAN controller approach can not guarantee low latencies for more than 2000 devices.

by extending the current programming libraries of MARC without altering the benchmark's system itself.

Furthermore, we discuss the validity of our benchmarking tool considering the UE and SD-RAN agent emulation. We stress that the results measured by MARC with respect to CPU, memory utilization and signaling overhead are in line with the results presented for FlexRAN [8] and 5G-EmPOWER [9] for a small amount of agents and UEs utilizing hardware, demonstrating credibility for our measurements.

In our work, we evaluated FlexRAN and 5G-EmPOWER for a monitoring application. Nonetheless, considering the capabilities of the aforementioned controllers and the necessity for more complex operations in 5G/6G networks, evaluations with respect to applications such as scheduling and RAN slicing become crucial. In this section, we discuss whether MARC is suitable for providing extensibility for such scenarios and applications. Generally, a real implementation of an SD-RAN controller for performing tasks such as scheduling or RAN slicing requires information with respect to the user characteristics. MARC constitutes the basis to enable such complex tasks by introducing the periodic report messages as an input to the controllers. Thus, if applicable by the SD-RAN controllers, our benchmarking tool can provide measurements similarly to the monitoring use-case.

Here, however we stress that due to the high complexity of such tasks, the results provided for the monitoring application may vary and a fewer number of UEs and BSs might be supported by those controllers. Following the same logic, the improvements introduced by AltFlexRAN with respect to the **RIB updater** and **task manager** will vary. In that regard, a more dynamic adaptation between the resources allocated for either **RIB updater** or **task manager** is mandatory to achieve better performance. Alternatively, variations in the presented results for FlexRAN could be observed when testing the recent version introduced by Mosaic5G [35]. Given the improved API [48], the amount and rate of statistics delegated to the SD-RAN controller can be adapted at runtime according to the wireless channel variations. Utilizing sophisticated algorithms this could lead to serving more BSs and UEs similar to AltFlexRAN. Results for the recent FlexRAN version [35] are not presented in this work, however, it remains an interesting area of research, which can be enabled by MARC due to its compatibility with the existing FlexRAN version [8] already evaluated. That said, MARC provides the basic setup for further research in the SD-RAN domain.

Finally, we highlight the stable operational region of the considered SD-RAN controllers in Table II. Notably, the single SD-RAN controller approach can not sustain the network load in an urban dense scenario (i.e., $>$ 5000 devices). Thus a distributed control plane becomes a must. However,

while this approach may balance the load of the network, it poses significant design challenges since it requires extra signaling overhead for the inter-controller communication. This nonetheless constitutes an interesting research topic for future work.

VII. Conclusion

In this paper we present MARC, a novel benchmarking tool for SD-RAN platforms and their controllers. We use MARC to measure and analyze FlexRAN and 5G-EmPOWER SD-RAN solutions considering metrics such as CPU, memory utilization, signaling overhead, and delay performance. In particular, our results are compared and validated with the results provided with hardware for a small amount UEs and SD-RAN agents. Furthermore, our analysis is extended to test the scalability of SD-RAN controllers in realistic network dimensions with up to 5000 devices. Based on our tool, we provide useful guidelines for the operation of SD-RAN controllers and propose solutions to overcome performance issues. Moreover, we highlight the scalability issue of single SD-RAN controller architectures and indicate the necessity for a distributed control plane based on multiple SD-RAN controller instances. We provide the full developed code of our benchmark to trigger further research in the context of SD-RAN controller development.

Acknowledgements

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - 438892507 and by a research grant from Zodiac Inflight Innovations (TriaGnoSys GmbH), Germany.

References

[1] E. J. Oughton, Z. Frias, S. van der Gaast, and R. van der Berg, "Assessing the capacity, coverage and cost of 5G infrastructure strategies: Analysis of the Netherlands," *Telematics and Informatics*, vol. 37, pp. 50–69, 2019.

[2] 3GPP, "TR 21.915 V1.1.0 (2019-03); Technical Report; Summary of Rel-15 Work Items (Release 15)," 2019.

[3] N. Alliance, "Description of network slicing concept," *NGMN 5G P*, vol. 1, 2016.

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

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

[6] X. Jin, L. E. Li, L. Vanbever, and J. Rexford, "Softcell: Scalable and flexible cellular core network architecture," in *Proceedings of the ninth ACM conference on Emerging networking experiments and technologies*, 2013, pp. 163–174.

[7] L. Cui, F. R. Yu, and Q. Yan, "When big data meets software-defined networking: SDN for big data and big data for SDN," *IEEE network*, vol. 30, no. 1, pp. 58–65, 2016.

[8] 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.

[9] 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.

[10] A. Papa, R. Durner, L. Goratti, T. Rasheed, and W. Kellerer, "Controlling Next-Generation Software-Defined RANs," *IEEE Communications Magazine*, vol. 58, no. 7, pp. 58–64, 2020.

[11] 3GPP, "TS 38.401 V15.6.0 (2019-07); Technical Specification Group Radio Access Network; Architecture description (Release 15)," 2019.

[12] S. Costanzo, I. Fajjari, N. Aitsaadi, and R. Langar, "A network slicing prototype for a flexible cloud radio access network," in *15th IEEE Annual Consumer Communications & Networking Conference*, 2018, pp. 1–4.

[13] N. Nikaein, X. Vasilakos, and A. Huang, "LL-MEC: Enabling low latency edge applications," in *IEEE 7th International Conference on Cloud Networking*, 2018, pp. 1–7.

[14] E. Coronado and R. Riggio, "Flow-based network slicing: Mapping the future mobile radio access networks," in *IEEE 28th International Conference on Computer Communication and Networks*, 2019, pp. 1–9.

[15] K. Ramantas, A. Antonopoulos, E. Kartsakli, P.-V. Mekikis, J. Vardakas, and C. Verikoukis, "A C-RAN based 5G platform with a fully virtualized, SDN controlled optical/wireless fronthaul," in *IEEE 20th International Conference on Transparent Optical Networks*, 2018, pp. 1–4.

[16] K. Koutlia, R. Ferrús, E. Coronado, R. Riggio, F. Casadevall, A. Umbert, and J. Pérez-Romero, "Design and experimental validation of a software-defined radio access network testbed with slicing support," *Wireless Communications and Mobile Computing*, 2019.

[17] A. Garcia-Saavedra, J. X. Salvat, X. Li, and X. Costa-Perez, "WizHaul: On the centralization degree of cloud RAN next generation fronthaul," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2452–2466, 2018.

[18] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "NVS: A substrate for virtualizing wireless resources in cellular networks," *IEEE/ACM transactions on networking*, vol. 20, no. 5, pp. 1333–1346, 2011.

[19] V. Sciancalepore, X. Costa-Perez, and A. Banchs, "RL-NSB: Reinforcement Learning-Based 5G Network Slice Broker," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[20] G. Garcia-Aviles, M. Gramaglia, P. Serrano, and A. Banchs, "POSENS: A practical open source solution for end-to-end network slicing," *IEEE Wireless Communications*, vol. 25, no. 5, pp. 30–37, 2018.

[21] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "vrain: A deep learning approach tailoring computing and radio resources in virtualized rans," in *The 25th Annual International Conference on Mobile Computing and Networking*, 2019, pp. 1–16.

[22] 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.

[23] 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.

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

[25] A. Papa. (2021) MARC. [Online]. Available: https://github.com/arled-papa/marc

[26] A. Gudipati, D. Perry, L. E. Li, and S. Katti, "SoftRAN: Software defined radio access network," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 25–30.

[27] M. Yang, Y. Li, D. Jin, L. Su, S. Ma, and L. Zeng, "OpenRAN: a software-defined ran architecture via virtualization," in *ACM SIGCOMM computer communication review*, vol. 43, no. 4, 2013, pp. 549–550.

[28] M. Y. Arslan, K. Sundaresan, and S. Rangarajan, "Software-defined networking in cellular radio access networks: potential and challenges," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 150–156, 2015.

[29] I. F. Akyildiz, P. Wang, and S.-C. Lin, "SoftAir: A software defined networking architecture for 5G wireless systems," *Computer Networks*, vol. 85, pp. 1–18, 2015.

[30] C. J. Bernardos, A. De La Oliva, P. Serrano, A. Banchs, L. M. Contreras, H. Jin, and J. C. Zúñiga, "An architecture for software defined wireless networking," *IEEE Wireless Communications*, vol. 21, no. 3, pp. 52–61, 2014.

[31] S. Katti and L. E. Li, "Radiovisor: A slicing plane for radio access networks," in *Presented as part of the Open Networking Summit 2014*, 2014.

[32] X. Foukas, M. K. Marina, and K. Kontovasilis, "Orion: Ran slicing for a flexible and cost-effective multi-service mobile network architecture," in *Proceedings of the 23rd annual international conference on mobile computing and networking*, 2017, pp. 127–140.

[33] 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.

[34] A. Papa, R. Durner, F. Edinger, and W. Kellerer, "SDRBench: A Software-Defined Radio Access Network Controller Benchmark," in *IEEE Conference on Network Softwarization (NetSoft), 3rd Workshop on Performance Issues in Virtualized Environments and Software Defined Networking (PVE-SDN)*, 2019.

[35] Mosaic5G. (2021) Flexran. [Online]. Available: https://mosaic5g.io/flexran/

[36] R. Sherwood and K.-K. Yap. CBench Controller Benchmarker. [Online]. Available: http://www.openflowswitch.org/wk/index.php/Oflops,2011

[37] M. Jarschel, F. Lehrieder, Z. Magyari, and R. Pries, "A flexible OpenFlow-controller benchmark," in *IEEE European Workshop on Software Defined Networking*, 2012, pp. 48–53.

[38] Open Networking Foundation. Libfluid. [Online]. Available: https://opennetworkingfoundation.github.io/libfluid/md_doc_Benchmarks.html

[39] A. Blenk, A. Basta, W. Kellerer, and S. Schmid, "On the impact of the network hypervisor on virtual network performance," in *IEEE/IFIP Networking Conference*, 2019, pp. 1–9.

[40] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, 2008.

[41] Python. (2019) Asynchronous I/O. [Online]. Available: https://docs.python.org/3/library/asyncio.html

[42] Google. (2019) Protocol buffers. [Online]. Available: https://developers.google.com/protocol-buffers/

[43] Python. (2019) Construct libary. [Online]. Available: https://pypi.org/project/construct/

[44] 3GPP, "TS 22.261 V16.8.0 (2019-06); Technical Specification Group Service and System Aspects; Service requirements for the 5G systems (Release 16)," 2019.

[45] Python. (2019) psutil libary. [Online]. Available: https://pypi.org/project/psutil/

[46] X. Foukas. (2016) Flexran-rtc. [Online]. Available: https://github.com/xfoukas/flexran-sdran

[47] R. Riggio. (2019) Empower Runtime. [Online]. Available: https://github.com/5g-empower/empower-runtime

[48] Mosaic5G. (2020) Flexran northbound api documentation. [Online]. Available: https://mosaic5g.io/apidocs/flexran/#api-Stats-SetStatsConf

Arled Papa completed his Bachelor of Science in Electronics Engineering at the Polytechnic University of Tirana, Albania in 2015. He received his Master of Science in Communications Engineering at the Technical University of Munich (TUM) in November 2017 with high distinction. In February 2018 he joined the Chair of Communication Networks at TUM as a research and teaching associate. His research focuses on the design and analysis of QoS, Network Slicing and Programmability of Sofware-Defined Radio Access Networks.

Raphael Durner received the M.Sc. degree in electrical engineering from the Technical University of Munich, Munich, Germany, in 2014. In November 2014, he joined the Chair of Communication Networks, Technical University of Munich as a Member of the Research and Teaching Staff, completing his Ph.D. in 2020. His research focuses on network softwarization approaches and their impacts on network performance.

Endri Goshi completed his Bachelor of Science in Electronics Engineering at the Polytechnic University of Tirana, Albania in July 2015. In October 2016, he joined the Master of Science in Communications Engineering program at the Technical University of Munich (TUM) and graduated in May 2019 with distinction. In July 2019 he joined the Chair of Communication Networks at TUM as a research and teaching associate. His research interests include Virtualized and Cloud-Native Mobile Core Networks, Software-Defined Networks and Programmable Networks.

Leonardo Goratti received his Ph.D. degree in wireless communications in 2011 from the University of Oulu, Finland, and his M.Sc. in telecommunications engineering in 2002 from the University of Firenze, Italy. He is currently part of research staff at Safran S. A.

Tinku Rasheed is a senior research staff member at Safran S. A. He has extensive industrial and academic research experience in the areas of mobile wireless communication and data technologies, end-to-end network architectures and services. He has several granted patents and has published his research in major journals and conferences.

Andreas Blenk received the Dr.-Ing. degree (Ph.D.) from the Technical University of Munich in 2018 with distinction. He joined the Chair of Communication Networks at the Technical University of Munich in June 2012, where he is currently working as a senior researcher and associate lecturer. Since March 2019, he is also a senior research fellow at the Communication Technologies group of the Faculty of Computer Science of the University of Vienna. His research is focused on self-driving, flexible and predictable virtual and software-defined networks, as well as data-driven network algorithms.

Wolfgang Kellerer (M'96, SM'11) received the Dr.-Ing. (Ph.D.) and Dipl.-Ing. (Master) degrees from the Technical University of Munich, Germany, in 1995 and 2002, respectively, where he is a Full Professor, heading the Chair of Communication Networks with the Department of Electrical and Computer Engineering. He was with NTT DOCOMOs European Research Laboratories for ten years in leading positions, contributing to research and standardization of LTE-A and 5G technologies. In 2001, he was a Visiting Researcher with the Information Systems Laboratory, Stanford University, CA, USA. His research has resulted in over 200 publications and 35 granted patents. He was awarded with an ERC Consolidator Grant from the European Commission for his research project FlexNets Quantifying Flexibility in Communication Networks in 2015. He currently serves as an Associate Editor for the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT and as an area editor for network virtualization for IEEE COMMUNICATIONS SURVEYS AND TUTORIALS. He is a member of ACM and VDE ITG.