FlexVRAN: A Flexible Controller for Virtualized RAN over Heterogeneous Deployments

Robert Schmidt, Chia-Yu Chang, Navid Nikaein
Communication Systems Department, EURECOM, France
Email: {robert.schmidt, chia-yu.chang, navid.nikaein}@eurecom.fr

***Abstract*—Alongside the mobile network evolution toward the fifth generation (5G) era, it is expected that the radio access network (RAN) will be the most challenging technology domain to serve multiple service requirements. Specifically, three critical aspects are particularly emphasized: (i) heterogeneous RAN deployments, (ii) RAN functional splits between disaggregated entities, and (iii) sliced RAN for multiple services. To synthesize these three different aspects, a unified and customizable control framework is needed to serve both needs of infrastructure provider and slice owner. To this end, we propose the FlexVRAN control framework as an extension to our previous work to provide a two-level abstraction scheme between the underlying physical infrastructures, logical base stations (BSs), and slice-specific virtual BSs. We present a proof-of-concept prototype of the proposed FlexVRAN over the OpenAirInterface (OAI) and FlexRAN platforms, and the evaluation results show the applicability and feasibility of software-defined RAN control over heterogeneous deployments in support of network slicing.**

I. Introduction

Several key technologies are widely considered toward 5G, e.g., millimeter-wave, network densification, and massive multiple-input multiple-output [1], incurring new challenges. For instance, network densification seems promising to improve area traffic capacity and connection density; however, the ways to coordinate densely-deployed base stations (BSs) and to efficiently utilize resource for multiple network services are still under study. To this end, the 5G architecture shall be designed with a certain level of flexibility via incorporating two essential principles: software-defined networking (SDN) and network function virtualization (NFV).

Based on the above two design principles, 5G will provide a paradigm shift to establish a flexible communication system. Unlike several previous generations, 5G not only provides a specific radio access technology (RAT) but also utilizes the available spectrum bands (e.g., millimeter-wave, sub-6 GHz) and access technologies (e.g., Wi-Fi, 4G, 5G) in a heterogeneous manner. In this sense, 5G can potentially fulfill diverse service requirements by flexibly hosting customized end-to-end networks using cloud and edge infrastructures, physical and virtual network functions (PNFs/VNFs) across various domains such as the radio access network (RAN) and the core network (CN).

Especially regarding the RAN, two critical points need particular investigation to achieve the required flexibility and programmability. First, the network shall be used efficiently and independently via crafting a logically separated space for each service, termed RAN slicing [2]. Second, each separated network can flexibly centralize its RAN processing at the centralized/edge cloud to compose the BS functionalities among disaggregated RAN entities (i.e., centralized unit (CU), distributed unit (DU), radio unit (RU)), termed as the Cloud RAN (C-RAN) [3], according to the functional splits as documented in 3GPP TR 38.801.

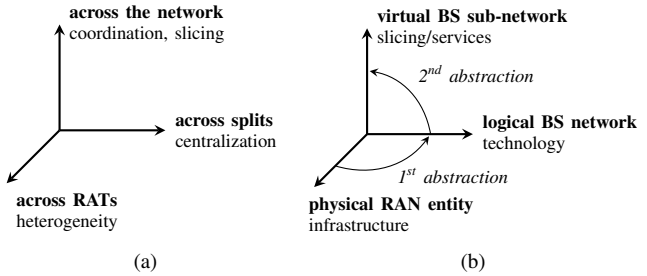

Figure 1. The 5G network can be deconstructed into various dimensions: (a) a RAT-split-network view, generalizing into (b) the physical, the logical, and the virtual dimensions.

Based on these observations, we identify three dimensions that the future 5G RAN will have to serve, depicted in Fig. 1a. Nevertheless, the control complexity and efficiency over the RAN domain across these three dimensions is still an open issue. Therefore, we envision a RAN controller that can bridge these dimensions by unifying different infrastructures, RATs and slices into a common network, while also revealing slice-specific sub-network views. As shown in Fig. 1b, the physical infrastructure using functional splits and various RATs can be abstracted to compose a logical base station (lBS) spanning the full protocol stack, e.g., 4G eNB or 5G gNB. Such an lBS can be seen as an enabler for effective management and control of the network by reducing network complexity of heterogeneous deployments. This lBS is further abstracted to provide a virtualized slice-specific sub-network for network slices, while still maintaining user plane programmability. Such a virtual sub-network is formed from a set of virtual BS (vBS) to reveal slice-specific resources and states serving the slice requirements. In total, this two-level abstraction can not only unify the disaggregated RAN control for network operators but also customize the service deployment for slice owners.

In summary, this work proposes a novel controller for virtualized RAN, FlexVRAN, to offer a unified and customizable RAN service environment considering heterogeneous deployments (e.g., distributed, disaggregrated, centralized) in multi-vendor, multi-RAT environments. Moreover, the aforementioned two-level abstraction approach can be provided

by FlexVRAN from disaggregated RAN entities, to lBSs, and further to vBSs forming a virtual sub-network. To this end, our contributions in this work are summarized as:

- Introduce a two level abstractions (logical and virtual) for a heterogeneous RAN that is applicable to current (LTE) and future (5G New Radio) RANs, and
- Present a realizable RAN controller design to support BS slicing through virtualization and customization,
- Provide a concrete implementation of FlexVRAN over the OpenAirInterface (OAI) LTE [4] and FlexRAN [5] platforms along with both system performance realization and use-case demonstration.

The paper is organized as follows: In Section II, we review the related work and highlight the goals of FlexVRAN. In Section III, we describe the proposed FlexVRAN system, and elaborate on the design details in Section IV. The implementation is assessed in Section V, and we conclude the paper in Section VI.

II. Background and State of the Art

In correspondence to the aforementioned 5G RAN evolution, a unified and customized control is necessary for the RAN domain. Such control scheme is essential to provide an agile RAN programmability over heterogeneous and disaggregated RAN entities for each network slice. In practice, two key enablers are highlighted: RAN virtualization, and software-defined RAN (SD-RAN). The former allows to compose a virtualized RAN with high flexibility and scalability to serve multiple innovative services, while the latter can decouple the control plane (CP) processing from the user plane (UP) processing to facilitate the customized RAN slicing.

RAN virtualization stems from the NFV technique to allow multiple vBSs to share common resources for multiple tenancies. For instance, the work of [6] provides the functional isolation in terms of the customized and dedicated CP functionalities for each mobile virtual network operator. In [7], a slice-based "network store" architecture is proposed as a platform to facilitate the dynamic network slicing based on the VNFs on top of underlay infrastructures. Moreover, the deployment of virtualized RAN shall consider the disaggregated resource. This RAN disaggregation is surveyed by the open network foundation (ONF) to provide the virtualization and slicing out of many disaggregated components. As highlighted in [8], the RAN disaggregation can offer the potentials of flexibility, scalability, upgradability and sustainability via the RAN service chaining notion.

As the second enabler, the SD-RAN concept is studied in several works arguing the level of centralization of CP functionalities. Fully centralized architectures are proposed in OpenRAN [9] and SoftAir [10] that may face the challenge of real-time control given the inherent delay between the controller and the underlying RAN. The SoftRAN architecture [11] statically refactors the control functions into the centralized and distributed ones based on the time criticality and the required central view. Moreover, the SoftMobile approach [12] abstracts these CP processing across network layers based on their functionalities and can perform the control functionalities through the application programming interfaces (APIs). Furthermore, FlexRAN [5] realizes an SD-RAN platform and implements a customized RAN southbound API through which programmable control logic can be enforced with different levels of centralization, either by the controller or RAN agent.

Orion [13] introduces the BS hypervisor to simultaneously isolate slice-specific control logics and share the radio resources. Moreover, the underlying PRBs are grouped into vRBs to be provided only to the corresponding slice. It exploits the prerequisites of function isolation and resource virtualization, while it does not consider a common slicing control system to support multi-cell coordination across different slices (e.g., in case of conflict) and different RAN deployment scenarios (e.g., monolithic and disaggregated), nor it provides control over the network topology to infrastructure owners.

We can observe that these above studies only focus on a part of enablers. To this end, the proposed FlexVRAN aims to provide a controller that exposes different levels of monitoring, control and coordination for infrastructure and slice owners using a common description for BSs along physical, logical, and virtual dimensions.

III. Proposed FlexVRAN Control Framework

Based on the aforementioned objective of FlexVRAN, we show how it can achieve the two-level abstraction from physical deployment over logical to virtual base stations in order to form a virtual sub-network that is revealed to a slice. For our purposes, we define a base station descriptor (BSD), whether it is physical, logical or virtual, as a triple of:

Resources describing radio spectrum resources which can be divided into bands, carriers, and physical resource blocks. For our purposes, we assume that all types of resources are provisioned by an orchestrator during the deployment phase.

Processing defining a set of functional blocks to perform CP/UP operations, separated through functional splits and described through *capabilities*, whether they are active, passive, or customized.

State is the status of the BS CP/UP processing and the associated configuration that are built up during runtime.

The FlexVRAN controller interacts with a number of split-aware RAN runtimes [2], which act as a local execution environment on top of each monolithic/disaggregated RAN entity. The underlying heterogeneous physical RAN entities host a number of RAN PNFs/VNFs for CP/UP processing[1]. They are annotated with (a part of) the BSD and the runtime exposes it to the FlexVRAN controller. Due to RAN disaggregation

[1]Some passive RAN entities do not possess the local RAN runtime due to their limited processing capabilities and therefore rely on the in-band control through the remote RAN runtime on top of other entities. For instance, as can be seen in Fig. 2a, the RU relies on in-band control through the DU. To this end, the operating functionalities and the relation toward other RAN entities for these RUs are maintained explicitly by the connecting DU. Also, since the RU is not activated without the DU, we assume the DU to possess the radio resources.

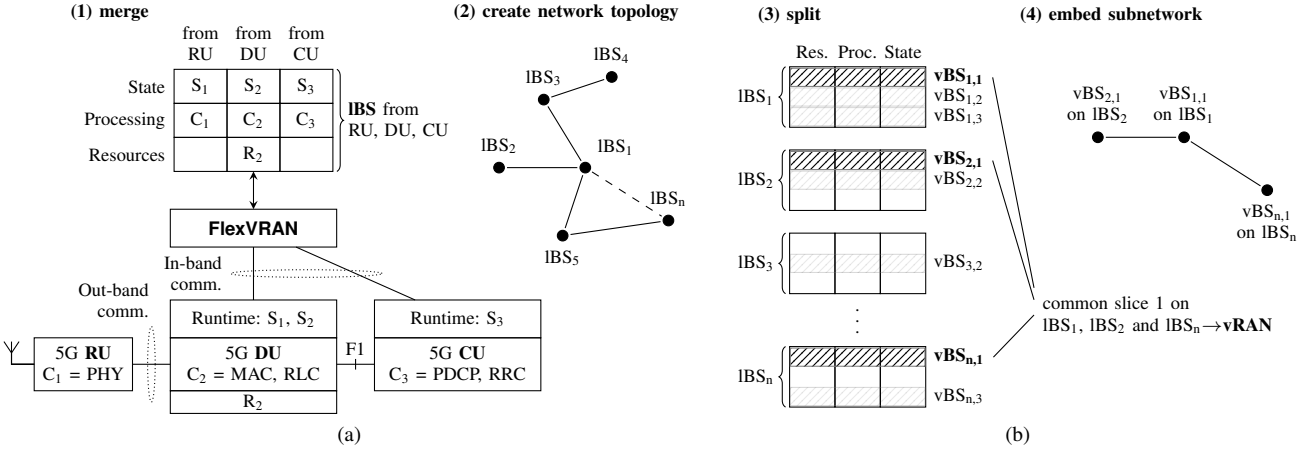

Figure 2. The operations of FlexVRAN to provide the (a) first-level abstraction with operations (1) and (2), and (b) second-level abstractions with operations (3) and (4). Note that the lBS triple in the second-level abstraction only represents customized state, processing, and resources, while common BS functionality remains in the lBS BSD. A missing topological relation between lBSs (dashed line) is present in the virtual sub-network.

coupled with heterogeneous deployments and technologies, it might however not be possible or feasible to provide a complete BSD of one BS. Hence, for the first-level abstraction from physical to logical BS, the FlexVRAN controller performs the operation of (1) merging BSDs of multiple RAN entities into one lBS and (2) creating a view of the network topology and annotating the lBS with the complete BSD, as shown in Fig. 2a. By matching capabilities of different RAN entities reflecting the actual chaining, i.e. if $\forall C_i, \bigcup_i C_i = C_{full}$ with C_{full} being the capabilities needed for an operational BS, the lBS is created as a unified representation of different deployments with a common data model. After this, the overall network topology is shown to represent the network cell structure in its spatial distribution.

To account for the heterogeneity of the underlying system, stemming from multi-vendor usecase-driven deployments, the BSD is represented as a unified data model like the network resource model defined by 3GPP in TS 28.541. The granularity of information in the BSD remains the same through merging. In fact, the first abstraction retains information granularity but consolidates the heterogeneity of the physical infrastructure (e.g., through functional splits) into mentioned data model (though certain operations like a reconfiguration of splits might need to expose information about splits, subject to access control). Due to this direct relation of the physical and logical representations, the infrastructure owner is able to perform a direct mapping between those two. At this stage, it is possible for the infrastructure owner to apply specific network operator control logic and operate FlexVRAN as an SD-RAN controller.

The FlexVRAN controller performs a second-level abstraction from lBS to vBS, creating a virtual sub-network specific to each slice owner. As shown in Fig. 2b, the controller performs the operation of (1) splitting (slice-wise customizable) lBSs into vBSs and (2) embedding them into the logical network topology in order to reveal a customized view of the virtual network (vRAN) to the slice owner. Splitting refers to storing slice-specific state and configuration, customizing functionality to tailor to the slice requirements, and possibly reserving resources exclusively for such a slice in the BSD of a vBS. This includes slice-specific control logic, to be enforced by the slice owner, described in the BSD of a vBS, which differs from the BSD of an lBS. Shared processing and state remain available read-only for all slices, e.g., monitoring, as long as there is no need for customization. Embedding on the other hand maps the vBSs within the topology of lBSs while decoupling it from the actual geographical position. This withholds network topology information from slice owners[2] and allows easier management of resource conflicts among BSDs. Thus, a slice owner might see a network of vBSs which do not expose the actual geographical position. For instance, vBSs might be moved from one lBS to the next following the user mobility pattern.

Through the capabilities, a slice owner is able to detect which processing functions exist and might customize slice-specific functionality, e.g., hand-over control logics. However, the information granularity, e.g., for specific system parameters, is adjusted depending on the specific SLA. For isolation purposes, the mapping of vBSs to lBSs (and hence to the physical infrastructure) or other slices is not possible from the slice owner's view. By revealing the embedded virtual sub-network, a slice can retain its service requirements via controlling its sub-network and the associated users.

Finally, one important task is to resolve conflicts occurring across different levels of abstraction. In general, the conflicts are resolved on the level that "creates" the corresponding level of abstraction.

- The slice owner can handle the slice-specific conflicts in its viewed vBSs.
- FlexVRAN is responsible for resolving the conflicts to compose an lBS, e.g., the proper chaining of RAN entities to compose available lBSs, and between slices when

[2]A slice owner might be an untrusted third party, or physical user location needs to be protected while verticals need to identify who uses their service.

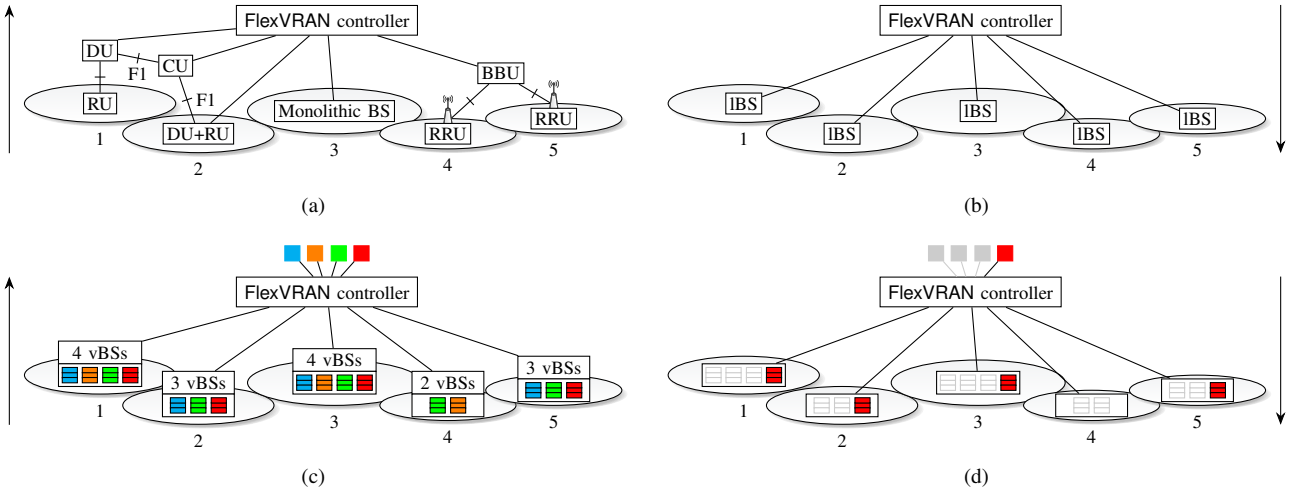

Figure 3. First- and second-level abstractions: (a) underlying RAN entities provided by the RAN runtime to the FlexVRAN controller, (b) the FlexVRAN controller exposes a number of lBSs, (c) each lBS hosts a number of vBSs belonging to different tenants, (d) one slice owner applies its control logic toward its viewed vBSs. The colors represent different slice owners.

customization attempts are made or new slices are to be instantiated.

- The RAN entity resolves conflicts among the underlying CP/UP processing, states and resources.

Nonetheless, there might be some conflicts that are not detectable by a given level; thus, the underlying level may reject or amend the control decisions. For instance, when the slice owner aims to configure the spectrum bandwidth of its vBS beyond the capability of the corresponding lBS, the FlexVRAN controller can override this control decision in the corresponding vBS configuration.

Fig. 3 represents the first- and second-level abstractions as created by FlexVRAN. As for the first-level abstraction, it merges the information of RAN entities into lBS BSDs and provides the topological information. The second-level abstraction further includes the RAN slicing notion by splitting lBS into vBS and embedding a virtualized and customized sub-network into the logical network topology for each slice owner.

IV. Design elements of FlexVRAN

FlexVRAN follows a hierarchical design suitable to realize real-time operation, and it is composed of a centralized controller and RAN runtime, one or two for each BS. The runtime might act as a local controller with a limited network view, handling control delegated by the controller, or in coordination with other runtimes and the controller.

A. RAN runtime

The RAN runtime [2] aims to provide a flexible execution environment to run multiple virtualized RAN instances with the requested levels of isolation and sharing of the underlying RAN modules for each slice instances. It can flexibly support various slice requirements (e.g., isolation) and elastically improve multiplexing benefits (e.g., sharing) in terms of (1) radio resource abstractions, (2) customized processing composition for modularized RAN, and (3) flexibility and adaptability to both monolithic and disaggregated deployments. On one hand, it allows the slice owner to create and manage slices, to perform customized control logic and CP/UP processing, to operate on a set of virtual resources or performance indicators, and to access the respective CP/UP states. On the other hand, it enables the infrastructure owner to efficiently and dynamically multiplex resources, processing, and states among multiple tenants to reduce the expenditures.

Moreover, the RAN runtime needs to provide several information to the FlexVRAN controller to compose the lBS. In practice, the RAN runtime provides its operator identity, underlying RAN entities' capabilities and RAT-specific information, supported user capability, the RAN runtime service capability, and so forth. The information from different RAN runtimes will be used to abstract the underlying physical infrastructure deployments into lBSs.

B. FlexVRAN controller

The FlexVRAN controller is the core element in our envisioned framework, and its components and interfaces are provided in Fig. 4. It has five main services for managing BS-specific or common information: (a) lBS manager, (b) virtualization manager, (c) task manager, (d) event notification service, and (e) conflict resolver.

First, the lBS manager is responsible to compose the lBS via interacting with the underlying RAN runtime instances through the south-bound interface by performing the operations of the first abstraction as described. For instance, it can merge the underlying RAN entities' capabilities to form an lBS comprising the baseband and protocol processing ranging from PHY up to RRC for both monolithic and disaggregated deployments. Note that the controller will withhold partial information of a BS and wait for the corresponding RAN runtime to connect. This abstraction is mainly used by the infrastructure provider gaining a simplified view on the RAN

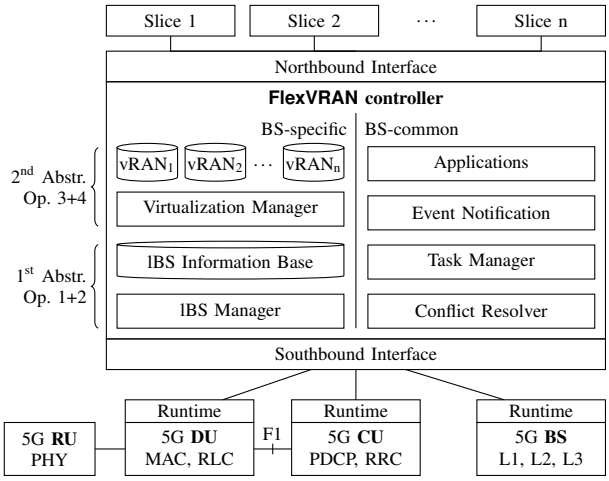

Figure 4. Components and interfaces of FlexVRAN controller.

including topological information. The formed lBSs will then be stored in the lBS information base and can be updated on-the-fly according to the dynamics of physical infrastructures and their capabilities.

Moreover, these lBSs can be treated as resources to be further abstracted as vBSs into vRANs based on the needs of the slice owner. This second-level abstraction is done by the virtualization manager, which abstracts the vRANs as virtual sub-networks based on slice owner's request. The northbound interface of the controller can restrict a slice's access to its vRAN, therefore isolating slices from each other through access control. Different levels of virtualization is supported by managing a slice context in the corresponding vRAN, such as the slice service level agreement (SLA) and customization configuration.

There are multiple common services provided by the FlexVRAN controller. In the first place, the task manager handles messages from the underlying RAN runtime instances. Via the southbound interface, the RAN runtime can update its capability and register itself toward the FlexVRAN controller together with the aforementioned information in Section IV-A. On the other hand, the task manager can perform basic create, read, update, and delete operations on the slice data from the provided network slice descriptor defining the required vBS, processing, resources, and states (as agreed between the slice owner and the network operator). When the slice admission control is finished, a slice can start to consume the services provided by the FlexVRAN control framework.

The event notification service provides updated information to managers and applications at the controller with the required level of granularity. It can happen in a periodic process with a larger window T than the underlying operating period, or on per event basis. For instance, when one physical infrastructure is deactivated, the corresponding life-cycle event will be notified toward the corresponding slice vBSs.

The conflict resolver provides a shared control logic for multiple slices. It aims to admit the customized control logic from different slice-specific control applications, resolve their conflicts, and enforce a feasible policy to underlying lBSs. For instance, two different slices may want to balance the load for its vBSs that are mapped from the same lBS; however, these two control decisions will incur a contradiction due to the limitation of the maximum number of supported users. To this end, the conflict resolver can either apply a policy-based method to decide which control application to be executed or use a learning-based method to generate a predictive model from the historical lBS state for decision making.

Finally, the FlexVRAN controller has both north- and south-bound interfaces toward each slice instances and the RAN runtime, respectively. In the south-bound, the protocol messages are exchanged through an asynchronous interface. Since the FlexVRAN controller has no a priori knowledge about the connected RAN runtime, RAN runtime information (identity and capability) is exchanged in between. The north-bound interface connects a slice to the FlexVRAN controller; through this, a slice runs in a separate process, either local or remote, and the interface provides a communication channel between a slice and its vRAN. Hence, each slice can be executed in isolation from each other either at the host or guest level leveraging well-known OS and virtualization technologies, such as containers or virtual machines.

V. Evaluation and Case Study

To validate the FlexVRAN control framework, we built a prototype of the FlexVRAN controller[3] supporting multiple RAN runtime instances based on the OpenAirInferface [4] and FlexRAN [5] platforms. In this work, we also developed the F1 functional split between the RLC and PDCP sub-layers, which is standardized by 3GPP in TS 38.470, in OAI[4]. The developed controller prototype supports both monolithic and disaggregated deployments. Our experiments are carried out using a USRP B210 software-defined radio connected to an x86 infrastructure, in which OAI-based RAN and CN are deployed with the FlexVRAN controller and RAN runtime. Also, we use commercial-off-the-shelf (COTS) Nexus 6P and Pixel 2 as user equipments (UEs) connected to the BS using frequency division duplex (FDD) mode with band 7.

A. Framework feasibility evaluation

First of all, we assess the feasibly of the proposed control framework by investigating the CPU and memory usage overhead at the RAN runtime, when connecting a BS under both monolithic and disaggregated deployment scenarios, toward the FlexVRAN controller in Fig. 5. These measurements of the RAN runtime overhead can justify the deployability of the RAN runtime together with the underlying RAN entities (i.e., BS, CU, or DU), while the FlexVRAN controller could be deployed on another physically separated x86 infrastructure. Moreover, we evaluate this overhead in two different scenarios: (1) zero connected UE (i.e., no user traffic), and (2) one connected UE saturating DL throughput with a video stream.

[3]Tag v2.1 of FlexRAN: https://gitlab.eurecom.fr/flexran/flexran-rtc

[4]F1 split in OAI develop: https://gitlab.eurecom.fr/oai/openairinterface5g/, measurements were done with commit 2f23047

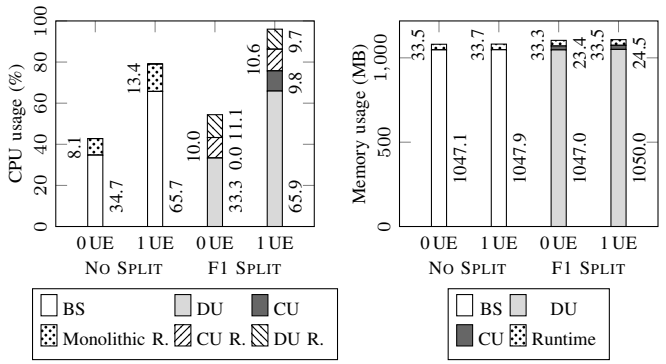

Figure 5. Comparison of OAI and FlexVRAN-enabled OAI. Note that the sum of memory usage of RAN runtimes is given for disaggregated deployment.

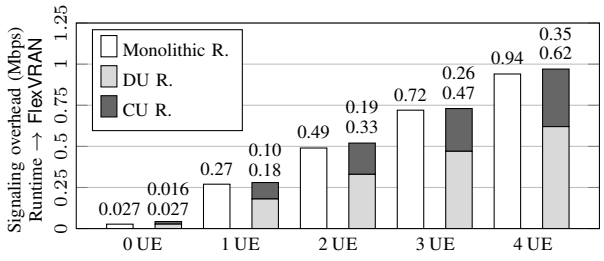

Figure 6. Signaling overhead from RAN runtime to FlexVRAN controller.

Further, the UE statistics and runtime configuration data to the controller were sent every 10 ms and 1 s, respectively.

We see that both monolithic (BS) and disaggregated (DU, CU) deployments have similar CPU usage. As a side note, we can observe that the extra processing for F1 interface of the disaggregated deployment is negligible. However, the two RAN runtimes of the disaggregated deployment (CU runtime and DU runtime) will both interact with the FlexVRAN controller, equally contributing to the CPU usage, whereas only one (BS) runtime is needed in the monolithic case.

As for the memory, we can first observe the overall memory usage of the disaggregated deployment (DU and CU) being slightly higher than the monolithic one (BS). The reason behind this is the extra memory overhead for the two RAN entities. Moreover, the RAN runtime introduces small overhead on the overall memory usage, which justifies the deployability of the RAN runtime over the underlying disaggregated RAN entity. Also, the number of connected UEs has little impact on the memory usage and it will not impact the applicability of the proposed FlexVRAN framework.

Furthermore, we investigate the signaling overhead for exchanging the control information between RAN runtime and FlexVRAN controller in Fig. 6, now sending statistics every 1 ms. Such signaling overhead is crucial when deploying the FlexVRAN control framework over disaggregated infrastructure with a capacity-limited fronthaul/midhaul network. We can see that the generated overhead mainly depends on the number of UEs in order to exchange user information periodically. However, the signaling overhead remains far lower than user plane traffic which might attain Gbps. For disaggregated deployments, the additional overhead is negligible.

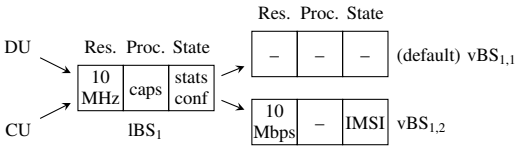

Figure 7. A BSD as merged from DU and CU for the small cell, then split into two vRANs (one default, one with resource reservation).

B. A slice owner's virtual sub-network through FlexVRAN

To illustrate the advantages of the controller framework especially for the slice owner, we consider a RAN network composed of two BSs in direct neighborhood using different operating frequencies in the same band. One BSs is a monolithic deployment as macro cell with a single runtime and 5 MHz bandwidth and the other is a small cell using the F1 functional split with each a runtime for DU and CU, operating on 10 MHz bandwidth. Both deployments serve one UE. All three runtimes are connected to the FlexVRAN controller which discovers two full capability sets and merges three RAN runtimes into two lBSs (cf. operation 1, Fig. 2a) with associated BSD. Further, FlexVRAN can build the topology of the network (cf. operation 2). More specifically, it only contains two unconnected lBSs due to inter-frequency deployment.

Regarding the deployed service, one slice owner aims to provide a video service with reserved resources for a specific event via requesting 10 Mbps of maximum guaranteed bit rate, and then associates its UEs to the corresponding vBS using their IMSIs. The applied FlexVRAN controller APIs are used as follows:

```
POST /install_vnetwork/10000000
POST /associate_ue_vnetwork/3  ↩
    --data-binary @imsi_list.json
```

The controller calculates the needed resource share, compares it with existing slices in both lBSs (each with a default slice that can be shrunk), and sends a command to the RAN runtimes to split the lBSs (cf. operation 3, Fig. 2b). The resulting vBS's BSD contains configuration of associated UEs in the state, with no customized processing and dedicated resources reserved for the requested bit rate. An example for such a vBS BSD split from a lBS BSD is given in Fig. 7. As discussed in Section III, an instantiation of slices in the runtime is subject to conflict management that might override a decision. If the instantiation succeeds, the controller embeds the sub-network into the previously built topology using two inter-connected vBSs to reflect that both vBSs serve instantiated video services in the same vRAN (cf. operation 4).

The reduction of control complexity through FlexVRAN controller is depicted in Fig. 8. A single read for the requested vRAN information can substitute multiple status read operations from RAN entities. Also, the following request write and optional check operations can largely reduce the control overhead of programming the RAN.

We measure UE performance based on the above experiment setup for throughput and delay jitter using iperf as well as latency using ping with 20 kB packets and an inter-

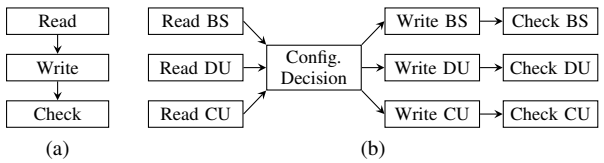

Figure 8. Reduced control complexity of slice creation: (a) With FlexVRAN (b) Without FlexVRAN.

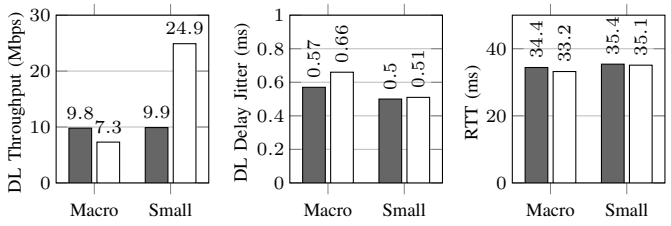

Figure 9. Comparison of a monolithic macro cell using 5 MHz and an F1-split small cell using 10 MHz. The dark bars show the performance of the UE in the video slice, the white ones show the UE using the default slice.

packet interval of 200 ms. The results shown in Fig. 9 indicate that the slice owner's requested throughput (dark bars) is almost achieved for both UEs for both deployment types. Deviations from the requested bit rate are mainly due to wireless channel variations as well as the fact that our current implementation rounds a bit rate to resource block groups, leading to "scheduled" bit rates that deviate from the requested bit rate. In both cases, similar jitter and delay performance are attained. Note the small additional delay overhead due to the additional hop between CU and DU. We highlight the fact that even with heterogeneous deployments, the proposed control framework can not only use one simple API call to instantiate the requested slice but also facilitate slice owners to customize and control their own virtualized sub-networks.

Finally, we show how a slice owner can control its slice through the vBS BSD. As shown in Fig. 10, the video slice owner requests a bit rate of 18 Mbps which is later increased to 22 Mbps, then 25 Mbps. Indeed, the requested bit rate is attained shortly after reconfiguration which confirms the applicability and feasibility of the proposed FlexVRAN control framework.

VI. Conclusions and Future work

In this work, we propose the unified and customized FlexVRAN control architecture framework to deal with the ever-evolving RAN deployments for multiple services. It provides a two-level abstraction scheme to serve the needs of both infrastructure providers and slice owners without introducing significant overhead in disaggregated and monolithic deployments. Finally, we implement its prototype over the OAI and FlexRAN platforms and show its feasibility.

In the future, we plan to extend the current work in the following directions: (1) a theoretical analysis of the abstractions, (2) an evaluation of "on-the-fly" change of functional splits, and (3) an investigation of the scheduling impacts of multiple services over the devised FlexVRAN control framework.

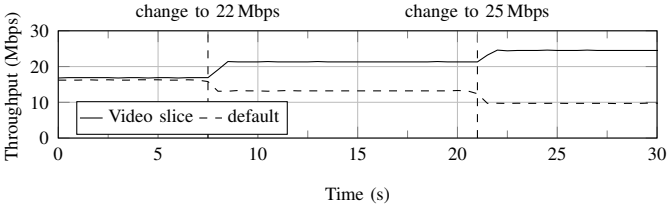

Figure 10. Network throughput after successive writes to the BSD of the corresponding vBS.

Acknowledgment

This work receives funding from the European Union's Horizon 2020 Framework Programme under grant agreement No. 762057 (5G-PICTURE) and No. 761913 (SliceNet).

References

[1] M. Shafi *et al.*, "5G: A tutorial overview of standards, trials, challenges, deployment, and practice," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 6, pp. 1201–1221, Jun. 2017.

[2] C.-Y. Chang and N. Nikaein, "RAN runtime slicing system for flexible and dynamic service execution environment," *IEEE Access*, vol. 6, pp. 34 018–34 042, Jun. 2018.

[3] A. Checko *et al.*, "Cloud RAN for mobile networks - A technology overview," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 405–426, Firstquarter 2015.

[4] N. Nikaein *et al.*, "OpenAirInterface: A flexible platform for 5G research," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 33–38, Oct. 2014.

[5] X. Foukas *et al.*, "FlexRAN: A flexible and programmable platform for software-defined radio access networks," in *Proc. of the 12th International on Conference on Emerging Networking EXperiments and Technologies (CoNEXT '16)*, Dec. 2016, pp. 427–441.

[6] Y. Zaki *et al.*, "LTE mobile network virtualization," *Mobile Networks and Applications*, vol. 16, no. 4, pp. 424–432, Aug. 2011.

[7] N. Nikaein *et al.*, "Network store: Exploring slicing in future 5G networks," in *Proc. of the 10th International Workshop on Mobility in the Evolving Internet Architecture (MobiArch'15)*, Sep. 2015, pp. 8–13.

[8] A. Tzanakaki *et al.*, "Optical networking interconnecting disaggregated compute resources: An enabler of the 5G vision," in *Proc. of 2017 International Conference on Optical Network Design and Modeling (ONDM)*, May 2017, pp. 1–6.

[9] M. Yang *et al.*, "OpenRAN: A software-defined RAN architecture via virtualization," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 549–550, Aug. 2013.

[10] I. F. Akyildiz *et al.*, "SoftAir: A software defined networking architecture for 5G wireless systems," *Computer Networks*, vol. 85, pp. 1–18, Jul. 2015.

[11] A. Gudipati *et al.*, "SoftRAN: Software defined radio access network," in *Proc. of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN '13)*, Aug. 2013, pp. 25–30.

[12] T. Chen *et al.*, "SoftMobile: Control evolution for future heterogeneous mobile networks," *IEEE Wireless Commun.*, vol. 21, no. 6, pp. 70–78, Dec. 2014.

[13] X. Foukas *et al.*, "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 (MobiCom '17)*, 2017, pp. 127–140.