Evaluating the Cloud-RAN architecture: functional splitting and switched Ethernet Xhaul

Andres F. Ocampo*†, Mah-Rukh Fida*, Ahmed Elmokashfi*, Haakon Bryhni*
* SimulaMet – Simula Metropolitan Center for Digital Engineering
† OsloMet – Oslo Metropolitan University
Pilestredet 52, N-0167 Oslo, Norway
Email:andres@simula.no

***Abstract*—The Cloud-RAN architecture is a key enabler to building future mobile networks in a flexible and cost-efficient way. For instance, switched Ethernet is a prime candidate for mobile transport networks (Xhaul), due to its flexibility, ubiquity, and cost-effectiveness. Understanding its performance under different network configurations would allow concluding about its appeal for Cloud-RAN. On the other hand, evaluating resource sharing mechanisms is relevant to put in place best solutions to host multiple virtual Base Band Units (vBBUs) into the same compute infrastructure. This paper assesses the feasibility of using a switched Ethernet Xhaul, by instantiating two vBBUs using different functional splits. Moreover, this paper evaluates two mechanisms for sharing network interface cards (NIC) in a general purpose server (GPS) hosting vBBUs. Our results point to a marginal performance degradation caused by the switched Ethernet Xhaul and the NIC sharing mechanisms. Such deviations could be seen from the increase in average and maximum Jitter and RTT results.**

***Index Terms*—Cloud radio access network (Cloud-RAN), Fronthaul, Ethernet Xhaul, functional splits, 5G, LTE**

I. Introduction

Unlike its predecessors, the fifth generation of mobile networks 5G, aims to cater for a wide spectrum of services with diverse requirements. Dense deployments of small cells seems to be the most likely network scenario to meet the requirements for 5G [1]. However, such scenarios of dense small cells make the traditional distributed radio access network architecture not economically feasible anymore. Therefore, flexible and cost-efficient mobile networks design is needed to realize the 5G vision. A way to meet the cost requirements for 5G, is to design the radio access network following a centralized cloud-based architecture (Cloud-RAN), leveraging virtualization and shared computing.

Despite its appeal as a key enabler for 5G, the Cloud-RAN architecture faces critical latency constrains and capacity requirements. Deploying a vBBUs in a central compute unit, imposes stringent latency constrains to the transport network (referred to as Fronthaul network), which connects the antennas and the vBBU. Although optical fiber solutions can meet these requirements, deploying dedicated fibers per cell would be extremely expensive for dense scenarios of small cells. One way to overcome this problem is to use switched Ethernet based Fronthaul. A switched Ethernet Fronthaul allows aggregating and multiplexing several BBUs into the same transport network. However, switched Ethernet networks bring latency, latency variation, and time-frequency synchronization issues that can be harmful for mobile services.

978-3-903176-31-7

To ease the strict bandwidth and latency requirements on the Fronthaul, the 3GPP proposed a set of functional splits of the BBU processing functions [2]. Processing part of the functions in a distributed unit (DU) close to the antennas, the central compute unit (CU) processes the remainder functions. The more functions instantiated in the DU the more delay would be afforded in the Fronthaul network. Introducing both functional split units DU and CU, redefines the mobile transport network segments and their requirements in terms of latency and capacity: the Fronthaul is the segment between the remote radio head (RRH. It is deployed at the antennas, which performs analog/digital conversion of the signal) and the DU; the Midhaul is the segment connecting the DU and the CU; and the Backhaul, which connects the Cloud-RAN with the packet core. These transport segments are referred to as mobile Crosshaul (Xhaul). This paper evaluates the CloudRAN architecture by aggregating two vBBUs into the same switched Ethernet Xhaul. In addition, this paper analyzes two mechanisms for NIC sharing at a centralized compute unit hosting the vBBUs. The remaining of the paper is organized as follows: while section II presents the related work, section III describes the deployed Cloud-RAN architecture and assumptions. Section IV describes the experiments and analyses the results. Finally, section V concludes the paper.

II. Related work

To host multiple vBBUs in a GPS, a virtualization environment provides the necessary features. Previous works [3], [4] suggest that linux containers (LXC) achieves better performance than virtual machines or Docker. However, there is still a need to study resource sharing among vBBUs in a virtualization environment. Apart from it, a debate has arisen over which BBU functional split is optimal. The study in [5] summarized the works implementing and analyzing the different functional splits and in [6], [7] the authors evaluated the individual performance of vBBUs with functional split 6 and 2, using a point to point Ethernet Fronthaul. Nonetheless, there is a need to

study the performance of such functional splits when multiple vBBUs share network and compute resources. Furthermore, Ethernet scheduling and preemption mechanisms are being studied [8], [9] to guarantee bounded and deterministic delay in the Fronthaul. However, further research is needed to evaluate the feasibility of such mechanisms when aggregating traffic from multiple BBUs with different functional splits.

III. Network Scenario

To assess the realization of Cloud-RAN architecture, we consider the network scenario depicted in Figure 1. This network scenario is composed of two LTE vBBUs, namely vBBU1 and vBBU2, respectively. We assume that both of these vBBUs belong to the same packet core (e.g., Evolve Packet Core - EPC, for LTE). In addition, a background traffic flow is included to evaluate resource sharing among vBBUs. Referred to as Anritsu, this traffic is generated by the Anritsu MT1000A Network Master Pro Tester [10] at the Midhaul network, and has as destination the CU. Finally, an external server represents the internet access from the mobile network. The main components in this network scenario are described below.

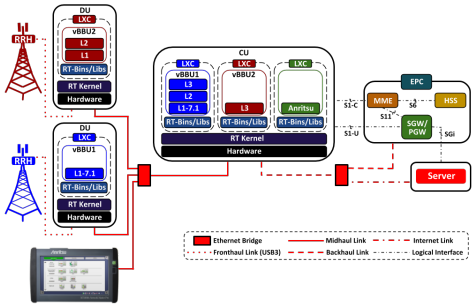


Fig. 1: Deployed Cloud-RAN architecture

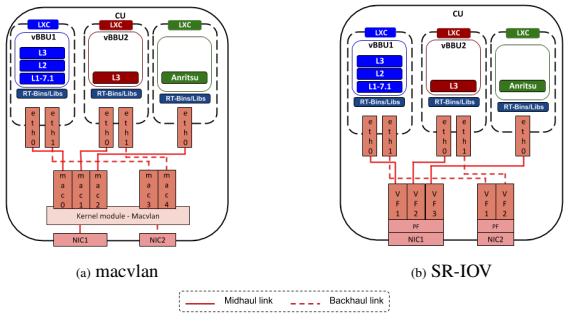
1) Deployed functional splits: vBBU1 adopts the functional split 7.1, which performs layer one (L1) low physical functions locally at the DU. The CU instantiates the remainder L1 functions, as well as layer two (L2) and layer three (L3) functions. Hence, split 7.1 interchanges frequency domain of I/Q samples between the DU and the CU, imposing stringent latency requirements in the Midhaul [5]. On the other hand, vBBU2 adopts the functional split 2. In this case, while the DU instantiates both L1 and L2 functions, the CU instantiates L3 functions. As split 2 hosts all real-time functions at the DU, the data rate requirement in the MidHaul are more relaxed and similar to that in the BackHaul.

2) Deployed virtualization environment: Each vBBU is deployed on top of a LXC at both the DU and the CU. In addition, a third LXC is deployed to receive the background traffic from the Anritsu.

3) Sharing the physical network adapter: The CU uses two NIC ports, which are shared among the LXCs. The first port, called NIC1, connects the CU to the Midhaul network. The second port, called NIC2, connects the CU to the Backhaul network. Consequently, each LXC hosting a vBBU deploys two virtual interfaces, namely eth0 and eth1. While eth0 is pinned to NIC1 - Midhaul network, eth1 is pinned to NIC2 - Backhaul network. On the other hand, the LXC dedicated to receive the Anritsu traffic instantiates a single virtual interface, namely eth0, which is pinned to NIC1. Sharing the NIC among the vBBUs might affect the overall network performance. For that reason, we study two NIC sharing mechanisms, as depicted in Figure 2:

Macvlan: Depicted in Figure 2a, macvlan allows creating virtual interfaces with their own MAC address and pinning them to a given NIC [11]. Through a hash procedure, packets are steered inside the kernel based on header's mac addresses. Three macvlan interfaces, namely mac0, mac1, and mac2, are instantiated on top of NIC1. These macvlan interfaces are assigned to the LXC's eth0, such as the LXC accesses the Midhaul as if it owns the NIC. Likewise, two macvlan interfaces, namely mac3 and mac4, are instantiated on top of NIC2. By assigning one macvlan to the LXC's eth1, the LXC accesses the Backhaul as if it owns the NIC.

SR-IOV: As depicted in Figure 2b, the Single-Root I/O Virtualization (SR-IOV) [12] allows sharing a NIC port, by virtualizing the PCI Express physical function (PF). To do so, a NIC's PF is partitioned into several virtual functions (VF), which can be allocated to guests in a virtualization environment. In this case, three VFs on NIC1, namely VF1,VF2, and VF3, are allocated to the LXC's eth0. This way, a LXC has isolated access to the Midhaul network. In addition, allocated to the LXC's eth1, two VFs on NIC2, namely VF3 and VF4, enable isolated access of the LXCs to the Backhaul network.



(a) macvlan (b) SR-IOV

Fig. 2: NIC sharing

4) Deploying Ethernet as mobile Xhaul: The Fronthaul is a point to point link, which transports time-domain IQ signal samples. The Midhaul, on the other hand, transports data which requirements depends upon the chosen functional split. An Ethernet switch aggregates the mobile traffic from both vBBUs along with the background traffic from the Anritsu, into a shared Midhaul link. In the Backhaul, a second Ethernet switch aggregates the Backhaul traffic from each vBBU, connecting them to the EPC.

IV. Experiments and Discussion

We evaluate the individual network performance for each of the vBBUs in the proposed network scenario, using following experimental setup.

A. Testbed setup

1) RRH: Both RRHs use the Ettus (B210) Universal Software Radio Peripheral (USRP) platform. These RRHs involve one antenna port in a Single Input Single Output (SISO) antenna array configuration.

2) DU: Both DUs use Intel NUC7i7BNB microcomputers, equipped with four Intel Core i7-7567U processors, and 32 GiB of memory. The host OS is Ubuntu 16.04 with low-latency Linux kernel version 4.19.58.

3) CU: The CU is a GPS equipped with sixteen Intel i7-8750H processors at 2.20 GHz, and 32 GiB of memory. The host OS is Ubuntu 18.04 with low-latency Linux kernel version 5.3.28. Moreover, the CU uses the Supermicro AOC-SG-i2 Gigabit Ethernet adapter, equipped with two Intel 82575 Gigabit Ethernet ports. According to the NIC sharing mechanisms described in section 3, while one of the ports instantiates the NIC1, the second port instantiates the NIC2.

4) Xhaul: For the Fronthaul link connecting the RRH with the DU, the B210 provides a fast SuperSpeed USB 3.0 connectivity at 5.0 Gbit/s. On the other hand, both the Midhaul network and the Backhaul network use the Juniper EX4200 Ethernet switch, with physical interfaces at 1 Gbit/s.

5) vBBUs: Based on their functional split, the vBBUs deploy the eNodeB-LTE implementation from OpenAirInterface (OAI). Particularly, in the case of vBBU1, OAI implements split 7.1 using the NGFI-IF4p5 interface specification [13]. In the case of vBBU2, OAI implements split 2 using the F1 Application Protocol (F1AP) [14], [15]. The wireless transmission setusp for both vBBUs, consists of 25 Physical Resource Blocks (PRB), which provides 5 MHz bandwidth.

6) UE: A single user equipment (UE) is connected to each vBBU. In both the cases, we used OnePlus-5 phones.

7) The EPC: The EPC deploys the 4G-5G OAI EPC, which runs on an independent GPS, equipped with four Intel i7 processor at 2.20GHz, and 12 GiB of memory.

B. Defining the baseline

We define baseline scenarios for each vBBU, as illustrated in Figure 3. For each baseline, a vBBU runs in a stand alone mode (i.e. no resource sharing among vBBUs at the CU, neither additional traffic flows). Also, the MidHaul and BackHaul use direct point to point links instead of Ethernet switches. This provides the baseline network performance for each vBBU.

C. Background traffic

When evaluating a given vBBU, we consider two cases of background traffic. First, a downstream UDP flow served by the second vBBU at a fixed rate of 13 Mbps. For example, if vBBU1 is being evaluated, the server outside the cellular network generates the UDP flow using iperf3 tool. This flow is destined to the UE connected to vBBU2. Optionally, the Anritsu network analyzer generates a traffic flow from the Midhaul network at a fixed rate of 850 Mbps and a fixed packet size of 1000 bytes. Destined to the designated LXC, this background traffic seeks to stress both NIC1, and the Midhaul network.

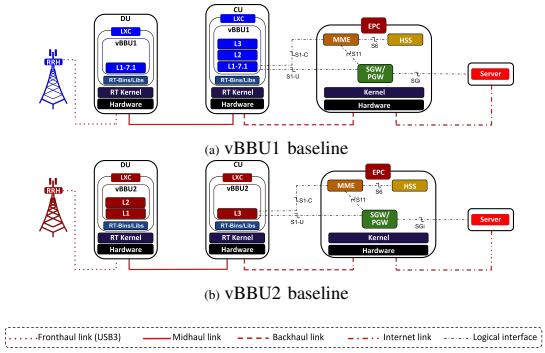


Fig. 3: Baseline scenarios

D. Conducted Experiments

1) Measuring Jitter in UDP downstream traffic: Using the Iperf3 tool, we benchmark a server-UE (client) data transmissions in terms of jitter. While the server outside the cellular network runs the Iperf server, the UE connected to the vBBU under evaluation runs the Iperf client. The test consists of sending downstream UDP flows from the server to the UE at different data rates.

2) Measuring the RTT: This test consists of sending ping-ICMP packets of different size from the UE, which is connected to the vBBU under evaluation, to the server outside the cellular network. Measured at the UE, RTT statistics from those packets provide insights into the end to end network latency.

E. Results

Following the notation sriov_a and sriov_noa, we represent the results for SR-IOV experiments with and without the Anritsu background traffic, respectively. Similarly, following the notation macvlan_a and macvlan_noa, we represent the results for macvlan experiments with and without the Anritsu background flow, respectively.

1) Results for vBBU1: Figure 4 shows the jitter as measured by the UE. For the baseline, the median jitter ranges from 0.5 msec for flows at 9 Mbps, to 0.7 msec for flows at 13 Mbps. In addition, the first and third quartiles exhibit moderate variations for all traffic flows, with a measured maximum value of 1 msec. For the proposed architecture, the median jitter increases by up to 0.3 msec for SR-IOV, and by up to 0.5 msec for macvlan, in comparison with the baseline reference values. Not only the median jitter increases, but the variance (based on the first and third quartiles) also increases. In some cases, the maximum values reach 1.7 msec, in comparison with the 1 msec reached by the baseline. Such an increase in the jitter is the result of using a switched Ethernet Xhaul network. Another reason is the CU's kernel network stack to process incoming packets. Macvlan has a higher median jitter and variance than SR-IOV. The reason is that the PCI virtualization mechanism provided by SR-IOV allows for a full traffic isolation and pass-through for the LXCs, avoiding any additional hashing procedure from the kernel as it does for macvlan.

RTT results are reported in Figure 5. In this Figure, while the data points represent the average RTT values for each

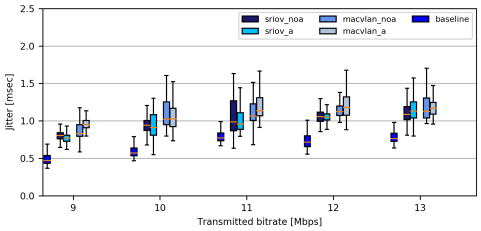


Fig. 4: Jitter measured at the UE for vBBU1

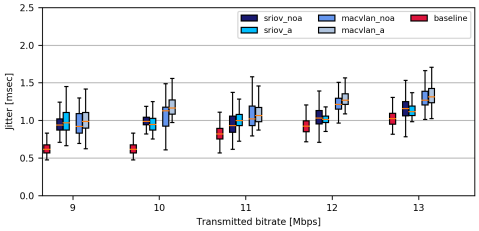


Fig. 6: Average Jitter measured at the UE for vBBU2

packet size, the whiskers indicate the standard deviation. In the baseline, the average RTT ranges from 33 msec to 35 msec, for packet sizes below 1000 bytes. For packet size of 1250 bytes and 1500 bytes, the average RTT increases to 36 msec and 37 msec, respectively. Moreover, the maximum standard deviation is 3 msec. In the case of macvlan, due to the switched Xhaul network and resource sharing at the CU, the standard deviation increases up to 5 msec. In contrast, because SR-IOV enables traffic isolation through PCI virtualization, the average RTT is 1 msec less than the baseline for almost all packet sizes. The standard deviation, though, increases up to 4 msec due to the switched Xhaul network and CU's resource sharing. Nonetheless, SR-IOV shows up to 2 msec lower average RTT than macvlan, that matters significantly for applications with stringent latency requirements.

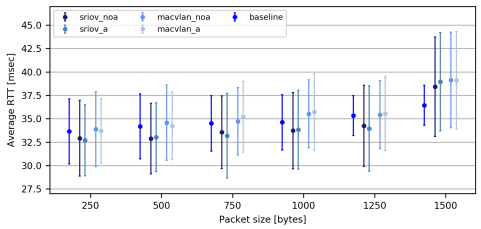


Fig. 5: Average Round-Trip Time measured at the UE for vBBU1

In general, with the traffic flow generated by Anritsu, the results remain stable. Even under such a congestion in the Midhaul and the CU, this architecture reveals the feasibility of running vBBUs on a shared architecture.

2) Results for vBBU2: As shown in Figure 6, the baseline median jitter ranges from 0.6 msec for flows at 9 Mbps, up to 1 msec for flows at 13 Mbps. Moreover, based on the first and third quartiles, the baseline exhibits moderate variation for all transmitted flows. In the proposed scenario, on the other hand, the median jitter increases by up to 0.3 msec for SR-IOV, and by up to 0.5 msec for macvlan, in comparison with the baseline results. Although the variance from the median is greater in the proposed scenario than in the baseline, SR-IOV shows more stable variance in comparison with macvlan. This supports the benefits of using a mechanism providing hardware based traffic isolation and pass-through from the NIC, as SR-IOV does.

Finally, figure 7 shows the results for the RTT experiments. In this case, the baseline shows average RTT values between 35 msec and 37 msec for packet sizes below 1500 bytes. For packets of 1500 bytes, the mean RTT increases up to 39 msec. While SR-IOV shows similar results as those of the baseline, macvlan shows higher average RTT values that are up to 1 msec more than the other two cases. In addition, the baseline and SR-IOV show similar standard deviation of 3 msec. Macvlan, conversely, shows a standard deviation of 3 msec to 5 msec higher than both the baseline or SR-IOV.

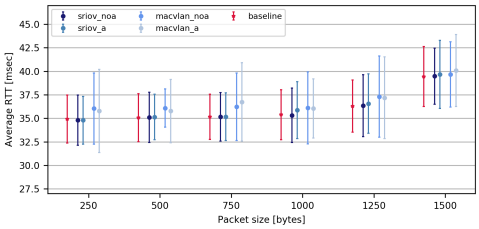


Fig. 7: Average Round-Trip Time measured at the UE for vBBU2

Although, in general, we observe a similar performance both for vBBU1 and vBBU2 compared to their baselines, the observed increase in jitter and RTT's standard deviation can be harmful for applications with stringent latency requirements. Our findings are in agreement with previous results that flag packet delay variation as a major concern when using switched Ethernet as Xhaul network [8]. This clearly creates a need for scheduling mechanisms that provides deterministic delay in the Xhaul.

V. Conclusions

This paper experimentally evaluates the Cloud-RAN architecture, where vBBUs implementing different functional splits share compute resources at the CU. Moreover, this paper evaluates the feasibility of using a switched Ethernet based Xhaul, such that vBBUs can be aggregated into the same Midhaul network. Our findings are encouraging showing that a switched Ethernet based Xhaul achieves similar RTT to that of standalone vBBUs, with a point-to-point connection between CU and DU. The switched Xhaul, however, suffers a higher median jitter and a higher variability in end to end RTT. We also find that NIC sharing mechanisms play an important role in determining the Cloud-RAN performance. In the future, we plan to evaluate scheduling mechanisms for providing deterministic and bounded delay over switched Ethernet Xhaul.

References

[1] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. T. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the

dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

[2] G. T. 38.801, "Study on new radio access technology: Radio access architecture and interfaces," 2017.

[3] N. Nikaein, E. Schiller, R. Favraud, R. Knopp, I. Alyafawi, and T. Braun, "Towards a cloud-native radio access network," in *Advances in Mobile Cloud Computing and Big Data in the 5G Era*. Springer, 2017, pp. 171–202.

[4] C.-N. Mao, M.-H. Huang, S. Padhy, S.-T. Wang, W.-C. Chung, Y.-C. Chung, and C.-H. Hsu, "Minimizing latency of real-time container cloud for software radio access networks," in *2015 IEEE 7th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE, 2015, pp. 611–616.

[5] L. M. Larsen, A. Checko, and H. L. Christiansen, "A survey of the functional splits proposed for 5g mobile crosshaul networks," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 146–172, 2018.

[6] N. Makris, P. Basaras, T. Korakis, N. Nikaein, and L. Tassiulas, "Experimental evaluation of functional splits for 5g cloud-rans," in *2017 IEEE International Conference on Communications (ICC)*. IEEE, 2017, pp. 1–6.

[7] G. Mountaser, M. Condoluci, T. Mahmoodi, M. Dohler, and I. Mings, "Cloud-ran in support of urllc," in *2017 IEEE Globecom Workshops (GC Wkshps)*, 2017, pp. 1–6.

[8] S. Bjørnstad, R. Veisllari, D. Chen, F. Tonini, and C. Raffaelli, "Minimizing delay and packet delay variation in switched 5g transport networks," *Journal of Optical Communications and Networking*, vol. 11, no. 4, pp. B49–B59, 2019.

[9] T. Wan and P. Ashwood-Smith, "A performance study of cpri over ethernet with ieee 802.1 qbu and 802.1 qbv enhancements," in *2015 IEEE global communications conference (globecom)*. IEEE, 2015, pp. 1–6.

[10] Anritsu, "Anritsu mt1000a network master pro tester," Last accessed Jul, 2020. [Online]. Available: https://www.anritsu.com/en-us/test-measurement/products/mt1000a

[11] J. Claassen, R. Koning, and P. Grosso, "Linux containers networking: Performance and scalability of kernel modules," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 713–717.

[12] Y. Dong, X. Yang, J. Li, G. Liao, K. Tian, and H. Guan, "High performance network virtualization with sr-iov," *Journal of Parallel and Distributed Computing*, vol. 72, no. 11, pp. 1471–1480, 2012.

[13] R. Knopp, "Overview of functional splits in oai," Last accessed February, 2020. [Online]. Available: https://www.openairinterface.org/docs/workshop/5_OAI_Workshop_20180620/KNOPP_OAI-functional-splits.pdf

[14] 3GPP, "5g; ng-ran; f1 application protocol (f1ap) (3gpp ts 38.473 version 15.2.1 release 15)," Last accessed May, 2020. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/138400_138499/138473/15.02.01_60/ts_138473v150201p.pdf

[15] OAI, "F1 interface," Last accessed Jul, 2020. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/f1-interface