Demo: Nuberu — A Reliable DU Design Suitable for Virtualization Platforms

Gines Garcia-Aviles
i2CAT Foundation

Andres Garcia-Saavedra
NEC Laboratories Europe

Marco Gramaglia
Universidad Carlos III de Madrid

Xavier Costa-Pérez
ICREA, i2CAT Foundation &
NEC Laboratories Europe

Pablo Serrano
Universidad Carlos III de Madrid

Albert Banchs
Universidad Carlos III de Madrid &
IMDEA Networks Institute

ABSTRACT

We demonstrate `Nuberu` [3]. The scenario consists of a DU under test (DuT), and one or more DUs sharing computing resources. A dashboard let us control (*i*) the type of DuT: "Baseline", implemented with vanilla `srsRAN` [5], or `Nuberu`; (*ii*) the number of competing vDUs; and (*iii*) their SNR. A second screen shows real-time metrics: (*i*) the processing latency of the TBs from each vDU instance; (*ii*) the throughput performance of DuT; (*iii*) the processing latency of DU jobs from DuT; and (*iv*) the ratio of latency constraint violations of DuT jobs. We show how the throughput attained by the baseline DU approach collapses upon sufficiently high computing interference from the competing DUs. Conversely, we show that the DU design introduced in [3] preserves reliability irrespective of the computing interference.

1 INTRODUCTION

The virtualization of radio access networks will become the spearhead of next-generation mobile systems beyond 5G [4, 7, 10]. Base stations are split into a central unit (CU), hosting the highest layers of the stack; a distributed unit (DU), hosting the physical layer (PHY); and a radio unit (RU), hosting basic radio functions such as amplification or sampling [1]. While CUs are amenable to virtualization in regional clouds, virtualized DUs (vDUs) require fast and predictable computing in edge clouds [2, 10].

Shared computing platforms provide a harsh environment for DUs because they trade off the predictability supplied by dedicated platforms for higher flexibility and cost-efficiency. Indeed, research has shown that contention in shared computing infrastructure lead to performance degradation compared to dedicated platforms [9, 11]—the so-called *noisy neighbor* problem. This is an issue for network functions such as virtual switches or even CUs, where metrics such as tail latency are relevant. However, the requirements associated with DUs are harder: violating deadlines cause users to lose synchronization, which leads to connectivity collapse.

2 THE SCENARIO

We use frequency division duplex where uplink (UL) and downlink (DL) transmissions occur concurrently in different

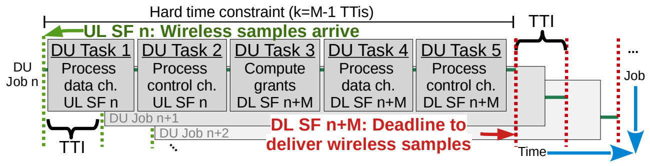
Hard time constraint (k=M-1 TTIs)
TTI
UL SF n: Wireless samples arrive
DU Job n
DU Task 1 Process data ch. UL SF n
DU Task 2 Process control ch. UL SF n
DU Task 3 Compute grants DL SF n+M
DU Task 4 Process data ch. DL SF n+M
DU Task 5 Process control ch. DL SF n+M
DU Job n+1
DU Job n+2
TTI
DL SF n+M: Deadline to deliver wireless samples
Time
Job
...

Figure 1: Every TTI (=1 ms), a worker executes a DU job, comprised of a pipeline of interdependent DU tasks to process UL SF n and DL SF $n + M$, within $M - 1$ ms.

frequency bands, and 5G's baseline numerology ($\mu = 0$ in 3GPP TS 38.211), which yields one transmission time interval (TTI) per subframe (SF), and a SF has a duration of 1 ms.

Fig. 1 illustrates the basic operation of the baseline DU processor [5, 6, 10]. Every TTI n, a **worker** initiates a **DU job** comprised of a *pipeline* of tasks (hereafter referred to as **DU tasks**): (1) process data and (2) control channels carried by UL SF n, (3) schedule UL/DL radio *grants* to be transported by DL SF $n+M$, and (4) process data and (5) control channels for DL SF $n + M$. A worker executes a DU job in a thread, using computing resources allocated by a task scheduler; and multiple workers perform DU jobs in parallel to handle one DL SF and one UL SF every TTI, as shown in Fig. 1.

Some operations in DU tasks, such as forward error correction, are compute-intensive and hence are implemented with highly-optimized libraries such as Intel FlexRAN [8]. Fig. 2 shows the time required by FlexRAN to decode large transport blocks (TBs) with different signal-to-noise-ratios (SNR) regimes and in a dedicated CPU core. Importantly, given 3GPP specification, there is a hard constraint on M that *imposes a computing time budget of roughly $M - 1$ ms to process each DU job* (usually, $M = 4$). See details in [3].

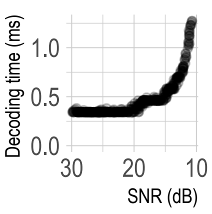

Figure 2: Decoding time of one TB in a dedicated CPU core.

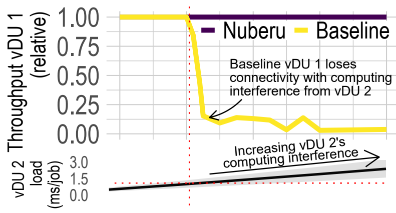

Figure 3: Two vDUs competing for computing resources. The UL/DL data load of the DuT is the highest possible while vDU 1's is variable.

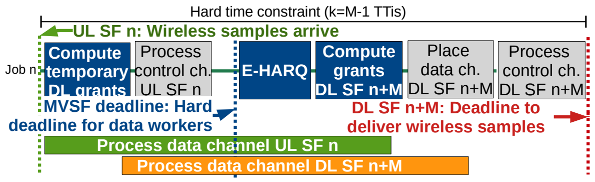
Hard time constraint (k=M-1 TTIs)
UL SF n: Wireless samples arrive
Job n
Compute temporary DL grants
Process control ch. UL SF n
E-HARQ
Compute grants DL SF n+M
Place data ch. DL SF n+M
Process control ch. DL SF n+M
MVSF deadline: Hard deadline for data workers
DL SF n+M: Deadline to deliver wireless samples
Process data channel UL SF n
Process data channel DL SF n+M

Figure 4: To provide reliability, `Nuberu` decouples data tasks from the rest of the pipeline by integrating a 2-stage DL scheduler, E-HARQ, and congestion control.

Violating this constraint has critical consequences on DUs. To illustrate this, we will present a scenario with one DU ($M = 4$), associated with one user implemented with `srsUE`, and virtualized over Linux containers sharing 5 Intel Xeon x86 cores @ 1.9GHz. We will refer to this vDU as DuT (Device-under-Test) and let it constantly transmit and receive large TBs with high SNR (the worst case in terms of computing workload). Concurrently, we will emulate the presence of additional vDUs (vDU 1, vDU 2, etc.) by deploying one instance of Intel FlexRAN per vDU processing TBs with configurable SNR over the same CPU pool. All the computing threads of the DuT and the competing vDUs have the same priority, and we use the default CPU scheduler of vanilla Ubuntu.

Our demonstrator consists of two terminals (T1 and T2). T1 will show a dashboard where we can control (*i*) the type of DuT: "Baseline", implemented with vanilla `srsRAN` [5], or `Nuberu`, which is our solution; (*ii*) the number of competing vDUs; and (*iii*) the SNR of each of these vDUs. T2 will show real-time metrics from the system. More specifically, T2 will show (*i*) The processing latency of the TBs from each vDU instance; (*ii*) The throughput performance of DuT; (*iii*) The processing latency of DU jobs from DuT; and (*iv*) the ratio of latency constraint violations of DuT jobs.

3 THE PROBLEM

We first select "Baseline" as DuT, and a single competing vDU (vDU 1) processing low-SNR TBs, which induce low computing interference. We then gradually increase its SNR, which in turn increases the competition with the DuT for computing resources. We will observe how this action incur higher latency when processing jobs for all the DUs in the system, including the DuT. Importantly, we will observe that when the ratio of latency constraint violations of the DuT become non-zero, the throughput of the DuT collapses.

Fig. 3 (top) depicts in yellow ("Baseline") that the DuT's throughput quickly deteriorates. The reason is that, because both vDUs share the same CPU pool, the DuT occasionally suffers from resource deficit when vDU 1's demand peaks. As a result, the DuT workers violate their deadline to send out a DL SF, which causes synchronization loss.

4 THE SOLUTION

`Nuberu` [3] is a novel pipeline architecture for 4G/5G DUs that is suitable for shared computing platforms. `Nuberu`'s design follows one objective: *to guarantee a minimum viable subframe (MVSF) for every TTI during moments of shortage in computing capacity to provide reliability and, provided this, maximize network throughput.* During such shortages, an MVSF encodes those signals and control information required to preserve user synchronization by temporarily holding off data delivery and relying on predictions.

To this end, there is a deadline within every DU job to begin building an MVSF even if data processing tasks are unfinished. This deadline, depicted in blue in Fig. 4, is set such that there is enough time to process an MVSF before the final job completion deadline (in red in the figure). To do this efficiently, data processing tasks need to be decoupled:

(1) To process DL data channel tasks:
- `Nuberu` adopts a two-stage DL scheduling approach:
  - *Temporary* DL grants are issued as early as possible in the pipeline, as shown in Fig. 4. Dedicated workers process (encode, modulate, etc.) these grants in threads and store the result in a buffer.
  - Upon the MVSF deadline, *final* DL data grants are computed based on those already processed (available in the buffer). Grants generated in a job n that are not processed on time are hence *delayed* for a later job.
- To mitigate the number of delayed DL data grants, the amount of DL data granted by the temporary scheduler is regulated by a *congestion controller* that adapts the flow to the availability of computing resources.

(2) To process UL data channel tasks:
- Dedicated workers process (demodulate, decode, etc.) UL data carried by each UL SF in separated threads.
- Upon the MVSF deadline, an *early HARQ* (E-HARQ) mechanism infers the *decodability* of UL data based on feedback from the workers, as shown in Fig. 4. This enables `Nuberu` to estimate the radio information that is required to build an MVSF even if UL data processing tasks have not finished on time.
- To maximize the performance of E-HARQ, which depends on the amount of work done before the MVSF deadline, another *congestion controller* adapts the UL data grants to the available computing capacity.

The details of `Nuberu` and a thorough experimental evaluation are presented in [3]. In this demonstration, we let the audience interact with the dashboard by m odifying the number of instances of competing vDUs and their SNR. In this way, we will demonstrate that `Nuberu` is resilient to computing capacity fluctuations, as shown by the purple line in Fig. 3, and hence is suitable for virtualization platforms.

ACKNOWLEDGEMENTS

This work has been supported by the EC through Grant No. 101017109 (DAEMON) and Grant No. 856709 (5Growth).

REFERENCES

[1] 3GPP. 2020. 5G;NG-RAN; Architecture description . 3GPP TS 38.401 version 16.2.0 Release 16.

[2] Cisco, Rakuten, Altiostar. 2019. Reimagining the End-to-End Mobile Network in the 5G Era. *White Paper* (2019).

[3] Gines Garcia-Aviles, Andres Garcia-Saavedra, Marco Gramaglia, Xavier Costa-Perez, Pablo Serrano, and Albert Banchs. 2021. Nuberu: Reliable RAN Virtualization in Shared Platforms. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking* (New Orleans, Louisiana) *(MobiCom '21)*. Association for Computing Machinery, New York, NY, USA, 749–761. https://doi.org/10.1145/3447993.3483266

[4] Andres Garcia-Saavedra and Xavier Costa-Perez. 2021. O-RAN: Disrupting the Virtualized RAN Ecosystem. *IEEE Communications Standards Magazine* (2021).

[5] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D Sutton, Pablo Serrano, Cristina Cano, and Doug J Leith. 2016. 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*. 25–32.

[6] Wang Tsu Han and Raymond Knopp. 2018. OpenAirInterface: A pipeline structure for 5G. In *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE, 1–4.

[7] Intel. 2017. vRAN: The Next Step in Network Transformation. *White Paper* (2017).

[8] Intel. 2019. *FlexRAN LTE and 5G NR FEC Software Development Kit Modules*. https://software.intel.com/content/www/us/en/develop/articles/flexran-lte-and-5g-nr-fec-software-development-kit-modules.html

[9] Antonis Manousis, Rahul Anand Sharma, Vyas Sekar, and Justine Sherry. 2020. Contention-Aware Performance Prediction For Virtualized Network Functions. In *Proceedings of ACM SIGCOMM '20* (Virtual Event, USA). ACM, 270–282.

[10] Samsung. 2019. Virtualized Radio Access Network: Architecture, Key technologies and Benefits. *Technical Report* (2019).

[11] Amin Tootoonchian, Aurojit Panda, Chang Lan, Melvin Walls, Katerina Argyraki, Sylvia Ratnasamy, and Scott Shenker. 2018. ResQ: Enabling SLOs in Network Function Virtualization. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, Renton, WA, 283–297. https://www.usenix.org/conference/nsdi18/presentation/tootoonchian