Processing Radio Access Network Functions in the Cloud: Critical Issues and Modeling

Navid Nikaein
EURECOM
navid.nikaein@eurecom.fr

ABSTRACT

Commoditization and virtualization of wireless networks are changing the economics of mobile networks to help network providers (e.g., MNO, MVNO) move from proprietary and bespoke hardware and software platforms toward an open, cost-effective, and flexible cellular ecosystem. Cloud radio access network is a novel architecture that perform the required base band and protocol processing on a centralized computing resources or a cloud infrastructure. This replaces traditional base stations with distributed (passive) radio elements with much smaller footprints than the traditional base station and a remote pool of base band units allowing for simpler network densification.

This paper investigates three critical issues for the cloudification of the current LTE/LTE-A radio access network. Extensive experimentations have been performed based on the OpenAirInterface simulators to characterise the base band processing time under different conditions. Based on the results, an accurate model is proposed to compute the total uplink and downlink processing load as a function of bandwidth, modulation and coding scheme, and virtualization platforms. The results also reveal the feasible virtualization approach towards a cloud-native radio access network.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design - Wireless Communication System.

Keywords

Cloud-RAN; LTE; BBU; Cloud Computing; Virtualization; RANaaS; OpenAirInterface.

1. FROM ANALOG TO VIRTUAL RADIO

In the last few decades, radio access networks (RANs) have significantly evolved from analog to digital signal processing units and from dedicated hardware components to reusable and flexible software-defined functions [1]. In a pure software-defined radio (SDR) system, the entire radio function runs on a general-propose processor (GPP) and only requires analog-to-digital and digital-to-analog conversions, power amplifiers, and antennas, whereas in typical cases, the system is based upon a programmable dedicated hardware (e.g. ASIC, ASIP, or DSP) and associated control software. Thus, the flexibility offered by a pure SDR improves service life-cycle and cross-platform portability at the cost of lower power and computational efficiency (i.e. ASIC: 1x, DSP: 10x, GPP: 100x).

Virtual RAN extends this flexibility through abstraction (or virtualization) of the execution environment. Consequently, radio functions become a general-purpose application that operates on top of a virtualized environment and interacts with physical resources either directly or through a full or partial hardware emulation layer. The resulted virtualized software radio application can be delivered as a service and managed through a cloud controller [2]. This changes the economics of mobile networks towards a cheap and easy to manage software platforms. Furthermore, cloud environment enables the creation of new services, such as RAN as a service (RANaaS) [3, 4], and more generally, network as a service (NaaS), such as LTEaaS, associated with the cloud RAN (C-RAN) [5–8]. C-RAN systems replace traditional base stations with distributed (passive) radio elements connected to a centralized baseband processing pool. Decoupling of the radio elements from the processing serves two main purposes. Centralized processing has the benefit of cost reduction due to fewer number of sites, easy software upgrade, performance improvement with coordinated multi-cell signal processing. Also, the remote radio heads have a much smaller footprint than a base station with on site processing, allowing for simpler and cost-effective network densification.

This paper analyses three critical issues in virtualization of the current LTE/LTE-A radio access network [9] and provides an accurate model of the LTE baseband processing load. Extensive experimentations have been performed based on the OpenAirInterface unitary simulators to accurately characterize the maximum processing time required to perform the LTE Rel.10 FDD baseband unit (BBU) functions. In contrast to the existing studies [10, 11], the impact of CPU architectures, individual BBU functions, SNR regime, and execution environments are also investigated under a realistic setting. Based on the results, an accurate model is proposed to compute the total uplink and downlink processing load as a function of number of physical resources blocks (PRB), modulation and coding scheme (MCS), and virtualization environments (VE). The results also reveal the feasible C-RAN architectures to enable a cloud-native radio access network.

The remainder of this paper is organized as follows. Section 2 investigates three critical issues in LTE FDD C-RAN. The evaluation setup and results are presented in Section 3. Section 4 provides the proposed modelling approach to compute the BBU processing. New directions and potential C-RAN architectures are elaborated in Section 5 and 6. Finally, concluding remarks are given in Section 7.

2. C-RAN CRITICAL ISSUES

While C-RAN comes with many attractive features and recent efforts have partially shown its feasibility, three critical issues need to be thoroughly investigated in order to assess the feasibility of C-RAN and identify the main design choices.

1. **Capacity requirement for fronthaul:** Because a typical BBU pool should support 10 - 1000 base stations, transport

of the I/Q samples from BBU to RRH requires a high fronthaul capacity. To meet the BBU timing requirements, fronthaul must provide an upperbound for the maximum one-way latency. Furthermore, clock synchronization across BBUs and RRH over the fronthaul also imposes a very low jitter.

2. **Latency requirements for BBU:** FDD LTE HARQ requires a round trip time (RTT) of 8ms that imposes an upper-bound for the sum of BBU processing time and the fronthaul transport latency.

3. **Real-time requirement for Operating System and virtualization environment:** Execution environment of BBU pool must provide (statistical) guarantee for the BBU pool successfully meeting their real-time deadlines related to the frame/subframe timing. It should also provide dynamic resource provisioning/sharing and load balancing to deal with the cell load variations.

In addition to above issues, C-RAN also brings many other challenges to BBU, RRH, and fronthaul [7]. Front-haul multiplexing and topology, optimal mapping (clustering) between BBUs and RRHs, efficient BBU interconnections, cooperative radio resource management, energy optimization and harvesting techniques, and channel estimation are just few examples.

The following subsections focus on the critical issues, and present C-RAN feasible architectures.

2.1 Fronthaul Capacity

Many factors contribute to the data rate of the fronthaul, which depends on the cell and fronthaul configurations. Equation 1 calculates the required data rate based on such configurations.

$$C_{fronthaul} = \underbrace{2 \cdot N \cdot M \cdot F \cdot W \cdot C}_{\text{cell configuration}} \cdot \underbrace{O \cdot K}_{\text{fronthaul configuration}} \quad (1)$$

where N is the number of receiving/transmitting (Tx/Rx) antenna ports, M is the number of sectors, F represents the sampling rate, W is the bit width of an I/Q symbol, C number of carrier components, O is the ratio of transport protocol and coding overheads, and K is the compression factor. The following table shows the required data rate for a simple set of configurations. An overall overhead is assumed to be 1.33, which takes into account the protocol overhead ratio of 16/15 and the line coding of 10/8 (CIPRI case). It can be seen that the fronthaul capacity heavily depends on the cell configuration and rapidly increases with the increase of sampling rate, number of antennas/sectors and component carriers.

Table 1: Fronthaul capacity for different configurations

BW	N	M	F	W	O	C	K	Rate
1.4MHz	1x1	1	1.92	16	1.33	1	1	81Mb/s
5MHz	1x1	1	7.68	16	1.33	1	1	326Mb/s
5MHz	2x2	1	7.68	16	1.33	1	1	653Mb/s
10MHz	4x4	1	15.36	16	1.33	1	1/2	1.3Gb/s
20MHz	1x1	1	30.72	16	1.33	1	1	1.3Gb/s
20MHz	2x2	3	30.72	16	1.33	1	1	7.85Gb/s
20MHz	4x4	3	30.72	16	1.33	1	1	15.6Gb/s

Further data rate reduction can be obtained by offloading the BBU functions to RRH. As shown in Figure 1, the function split can be done by decoupling the L3/L2 from the L1 (labelled 4), or part of the user processing from the L1 (labelled 3), or all user-specific from the cell processing (labelled 2), or antenna-specific from non-antenna processing (labelled 1), which is different for the Rx and Tx chain.

The trade-off has to be made between the available fronthaul capacity, complexity, and the resulted spectral efficiency. Regardless of different possibilities in BBU function split, the fronthaul should still maintain the latency requirement to meet the HARQ deadlines. NGMN adopts fronthaul maximum one-way latency of 250μ [9]. Different protocols have been standardized for the fronthaul, namely CPRI (common public radio interface) representing 4/5 of the market, OBSAI (Open Base Station Architecture Initiative) representing 1/5 of the market, and more recently the Open Radio Interface (ORI) initiated by NGMN and now by ETSI ISG (Industry Specification Group).

2.2 BBU Functions

Figure 1 illustrates the main RAN functions in both TX and RX spanning all the layers, which has to be evaluated to characterise the BBU processing time and assess the feasibility of a full GPP RAN. Since the main processing bottleneck resides in the physical layer, the scope of the analysis in this paper is limited to the BBU functions. From the figure, it can be observed that the overall processing is the sum of cell- and user-specific processing. The former only depends on the channel bandwidth and thus imposes a constant base processing load on the system, whereas the latter depends on the MCS and resource blocks allocated to users as well as SNR and channel conditions. The figure also shows the interfaces where the functional split could happen to offload the processing either to an accelerator or to a RRH.

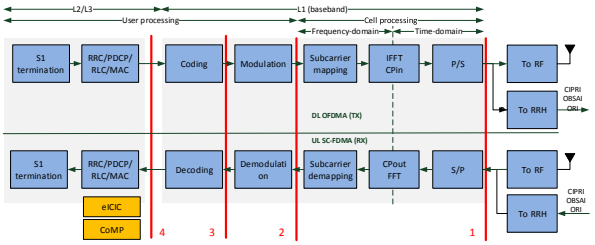
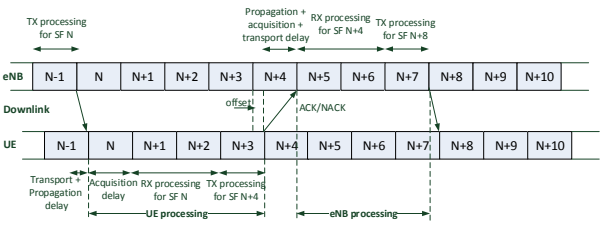


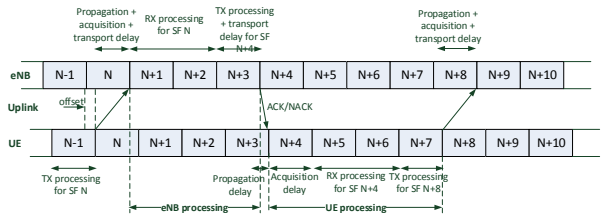
Figure 1: Functional block diagram of DL and UL for LTE eNB

To meet the timing and protocol requirements, the BBU processing must finish before the deadlines. One of the most critical processing that requires deadline is imposed by the Hybrid Automatic Repeat Request protocol (HARQ) in that every received MAC PDU has to be acknowledged (ACK'ed) or non-acknowledged (NACK'ed) back to the transmitter within the deadline. In FDD LTE, the HARQ Round Trip Time (RTT) is 8 ms. Each MAC PDU sent at subframe N is acquired in subframe $N+1$, and must be processed in both RX and TX chains before subframe $N+3$ allowing ACK/NACK to be transmitted in subframe $N+4$. On the receiver side, the transmitted ACK or NACK will be acquired in subframe $N+5$, and must be processed before subframe $N+7$, allowing the transmitter to retransmit or clear the MAC PDU sent in subframe N. Figure 2(a) and 2(b) show an example of timing deadlines required to process each subframe in downlink and uplink respectively.

It can be observed that the total processing time is 3ms, out of which 2ms is available for RX processing and 1ms for TX. Thus



(a) *DL HARQ timing*



(b) *UL HARQ timing*

Figure 2: FDD LTE timing

the available processing time for an eNB to perform the reception and transmission is upper-bounded by HARQ round trip time (T_{HARQ}), propagation time ($T_{Prop.}$), acquisition time ($T_{Acq.}$), and fronthaul transport time($T_{Trans.}$) as follows:

$$T_{rx} + T_{tx} \leq T_{HARQ}/2 - (T_{Prop.} + T_{Acq.} + T_{Trans.} + T_{Offset}) \quad (2)$$

where $T_{HARQ} = 8, T_{Prop.} + T_{Acq.} + T_{Trans.} + T_{Offset} \leq 1ms$, and $T_{offset} = 0$ in DL.

Depending on the implementation, the maximum tolerated transport latency depends on the eNodeB processing time and HARQ period. As mentioned earlier, NGMN adopted a 250 μs for the maximum one-way fronthaul transport latency. Hence, the length of a BBU-RRH link is limited within 20-40 km to avoid too high round-trip-delays (given that the speed of light in fiber is approximately 200 m/μs). At maximum distance of 15 km, the remaining overall processing time will be between 2.3–2.6 ms.

2.3 Real-time Operating System and Virtualization Environment

A typical general purpose operating systems (GPOS) is not designed to support real-time applications with hard deadline. Hard real-time applications have strict timing requirements to meet deadlines or otherwise unexpected behaviours can occur compromising the performance. For instance Linux is not a hard real-time operating system as the kernel can suspend a task when its runtime has completed and it can remain suspended for an arbitrarily long time. Kernel scheduling is the process in the OS that decides which task to run and allocates certain processing time to it. Such a scheduler is essential to guarantee the worst case performance and also to provide a deterministic behaviour (with short interrupt-response delay of 100 μs) for the real-time applications. Recently, a new scheduler, named SCHED_DEADLINE, is introduced in the Linux kernel mainstream that allows each application to set a triple $(runtime[ns], deadline[ns], period[ns])$, where $runtime \leq deadline \leq period$.[1] As a result, the scheduler preempts the kernel to meet the deadline and allocates the required runtime (i.e. CPU time) to each period.

Software-based Radio, is a real-time application that requires a hard deadlines to maintain the frame and subframe timing. In the C-RAN setting, the software radio application runs on a virtualized environment, where the hardware is either fully, partially, or not virtualized. Two main approaches exist to virtualization: virtual machines (e.g KVM[2] and Xen[3]) or containers (e.g. Linux Container LXC[4] and Docker[5]) as shown in Figure 3. In a virtual machine (VM), a complete operating system (guest OS) is used with the associated overhead due to emulating virtual hardware whereas containers use and share the OS and device drivers of the host. While VMs rely on the hypervisor to requests for CPU, memory, hard disk, network and other hardware resources, containers exploits the OS-level capabilities. Similar to VMs, containers preserve the advantage of virtualization in terms of flexibility (containerize a system or an application), resource provisioning, decoupling, management and scaling. Thus, containers are lightweights as they do not emulate a hardware layer (share the same kernel and thus application is native with respect to the host) and therefore have a smaller footprint than VMs, start up much faster, and offer near bar metal runtime performance. This comes at the expense of less isolation and greater dependency to the host's kernel.

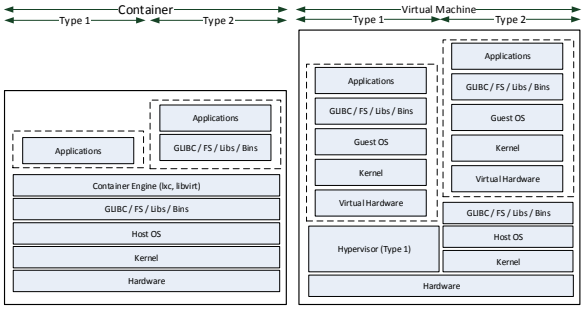


Figure 3: Comparison of a virtual machine and container virtualized environment.

Two other important aspects when targeting RAN virtualization are:

- **I/O Virtualization:** I/O access is a key for a fast access to the fronthaul interface and to the hardware accelerators that might be shared among BBUs. In hypervisor approach to virtualization (i.e. VM), IO virtualization is done through the hardware emulation layer under the control of hypervisor, where as in container this is done through the device mapping. Thus, direct access to the hardware is easier in containers than in VMs as they operate at the host OS level. In VM, additional techniques might be needed (e.g. paravirtualization or CPU-assisted virtualization) to provide a direct or fast access to the hardware. When it comes to sharing I/O resources among multiple physical/virtual servers, and in particular that of radio front-end hardware, new techniques such as multi root I/O virtualization (MR-IOV) are required.

- **Service composition of the software radio application:** A VBS can be defined as a composition of three types of ser-

[1] http://en.wikipedia.org/wiki/SCHED_DEADLINE

[2] http://www.linux-kvm.org

[3] http://www.xenserver.org/

[4] http://linuxcontainers.org

[5] http://www.docker.com

vice [3], atomic service that executes a single business or technical function and is not subject to further decomposition, composed service that aggregates and combines atomic services together with orchestration logic, and support service that provides specific (often common) functionalities available to all types of service. An atomic service in RAN can be defined on per carrier, per layer, per function basis. For instance, a VBS could be defined as a composition of layer 1 and layer2/3 services supported by a monitoring as a service.

3. EVALUATION

Four set of different experiments are performed. The first experiment analyses the impact of different x86 CPU architecture on BBU processing time, namely Intel Xeon E5-2690 v2 3Ghz (same architecture as IvyBridge), Intel SandyBridge i7-3930K at 3.20Ghz, and Intel Haswell i7-4770 3.40GHz. The second experiment shows how the BBU processing time scale with the CPU frequency. The third experiment benchmarks the BBU processing time in different virtualization environments including LXC, Docker, and KVM against a physical machine (GPP). The last experiment measures the I/O performance of virtual Ethernet interface through the guest-to-host round-trip time (RTT).

All the experiments are performed using the OpenAirInterface DLSCH and ULSCH simulators designed to perform all the baseband functionalities of an eNB for downlink and uplink as in a real system.All the machines (hosts or guests) operate on Ubuntu 14.04 with the low latency (LL) Linux kernel version 3.17, x86-64 architecture and GCC 4.7.3. To have a fair comparison, only one core is used across all the experiments with the CPU frequency scaling deactivated except for the second experiment.

The benchmarking results are obtained as a function of allocated physical resource blocks (PRBs), modulation and coding scheme (MCS), and the minimum SNR for the allocated MCS for 75% reliability across 4 rounds of HARQ. Note that the processing time of the turbo decoder depends on the number of iterations, which is channel-dependant. The choice of minimum SNR for a MCS represents the realistic behavior, and may increase number of turbo iterations and consequently causing high processing variation. Additionally, the experiments are performed at full data rate (from 0.6Mbps for MCS 0 to 64Mbps for MCS 28 in both directions) using a single user with no mobility, SISO mode with AWGN channel, and 8-bit log-likelihood ratios turbo decoder. Note that if multiple users are scheduled within the same subframe in downlink or uplink, the total processing depends on the allocated PRB and MCS, which is lower than a single user case with all PRBs and highest MCS. Thus, the single user case represents the worst case scenario.

The processing time of each signal processing module is calculated using timestamps at the beginning and at the end of each BBU function. OAI uses the `rdtsc` instruction implemented on all x86 and x64 processors to get a very precise timestamps, which counts the number of CPU clocks since reset. Therefore the processing time is measured as number a number of clocks when a function starts and ends divided by the CPU frequency.[6]

To allow a rigorous analysis, total and per function BBU processing time are measured.For statistical analysis, a large number of processing_time samples (10000) are collected for each BBU function to calculate the average, median, first quantile, third quantile, minimum and maximum processing time for all the subframes in uplink and downlink.

3.1 CPU Architecture Analysis

Figure 4 depicts the BBU processing budget in both directions for the considered Intel x86 CPU architecture. It can be observed that the processing load increases with the increase of PRB and MCS for all CPU architectures, and that it is mainly dominated by the uplink. Furthermore, the ratio and variation of downlink processing load to that of uplink also increases with the increase of PRB and MCS. Higher performance (lower processing time) is achieved by the Haswell architecture followed by SandyBridge and Xeon. This is primarily due to the respective clock frequency (c.f. Section 3.2, but also due to a better vector processing and faster single threaded performance of Haswell architecture.[7] For the Haswell architecture, the performance can be further increased by approximately a factor of two if AVX2 (256-bit SIMD compared to 128-bit SIMD) instructions are used to optimize the turbo decoding and FFT processing.

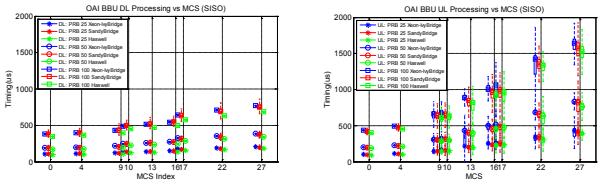


Figure 4: BBU processing budget in downlink (left) and uplink(right) for different CPU architecture.

3.2 CPU Frequency Analysis

Figure 5 illustrates the total BBU processing time as a function of different CPU frequencies (1.5, 1.9,2.3,2.7,3.0, and 3.4 GHz) on the Haswell architecture. The most time consuming scenario is considered with 100 PRBs and downlink and uplink MCS of 27. In order to perform experiments with different CPU frequencies, Linux ACPI interface and *cpufreq* tool are used to limit the CPU clock.It can be observed that the BBU processing time scales down with the increasing CPU frequency. The figure also reflects that the minimum required frequency for 1 CPU core to meet the HARQ deadline is 2.7GHz.

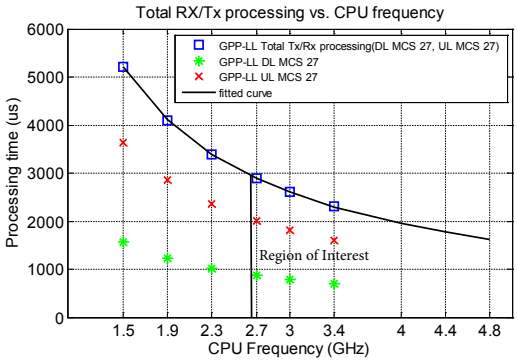


Figure 5: Total processing time as a function of CPU frequency.

Based on the above figure, the total processing time per subframe, $T_{subframe}$, can be modelled as a function of CPU frequency [11]:

[6] c.f. https://svn.eurecom.fr/openair4G/trunk/openair1/PHY/TOOLS/time_meas.h

[7] http://en.wikipedia.org/wiki/Haswell_(microarchitecture)

$$T_{\text{subframe}}(x)\ [\text{us}] = \alpha/x$$

, where $\alpha = 7810 \pm 15$ for the MCS of 27 in both directions, and x is CPU frequency measured in GHz.

3.3 Virtualization Technique Analysis

Figure 6 compares the BBU processing budget of a GPP platform with different virtualized environments, namely Linux Containers (LXC), Docker, and KVM, on the SandyBridge architecture(3.2GHz). While on average the processing time are very close for all the considered virtualization environments, it can be observed that GPP and LXC have slightly lower processing time variations than that of DOCKER and KVM, especially when PRB and MCS increase.

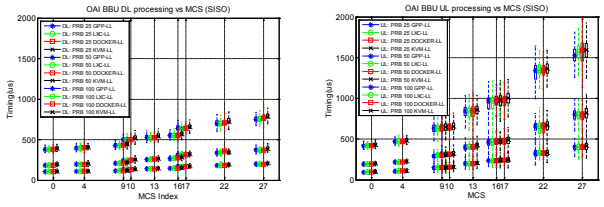


Figure 6: BBU processing budget in downlink (left) and uplink(right) for different virtualized environments.

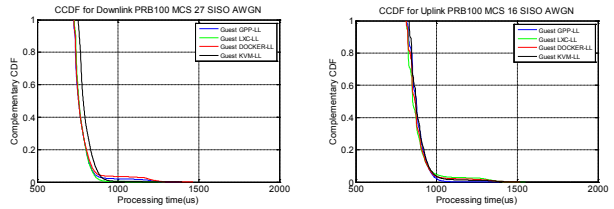


Figure 7: BBU processing time distribution for downlink MCS 27 and uplink MCS 16 with 100 PRB.

Figure 7 depicts the Complementary Cumulative Distribution Function (CCDF) of the overall processing time for downlink MCS 27 and uplink MCS 16 with 100 PRB. The CCDF plot for a given processing time value displays the fraction of subframes with execution times grater than that value. It can be seen that the execution time is stable for all the platforms in uplink and downlink. The processing time for the KVM (hypervisor-based) has a longer tail and mostly skewed to longer runs due to higher variations in the non-native execution environments (caused by the host and guest OS scheduler). Higher processing variability is observed on a public cloud with unpredictable behaviors, suggesting that cares have to be taken when targeting a shared cloud infrastructure [11].

3.4 I/O Performance Analysis

Generally, the one-way-delay of fronthaul depends on the physical medium, technology, and the deployment scenario. However in the cloud environment, the guest-to-host interface delay (usually Ethernet) has to be also considered to minimize the access to the RRH interface. To assess such a delay, bidirectional traffics are generated for different set of packet sizes (64, 768, 2048,4096,8092) and inter-departure time (1, 0.8, 0.4, 0.2) between the host and LXC, Docker, and KVM guests. It can be seen from Figure 8 that LXC and Docker are extremely efficient with 4-5 times lower round trip time. KVM has a high variations, and requires optimization to lower the interrupt response delay as well as host OS scheduling delay. The results validate the benefit of containerization for high performance networking.

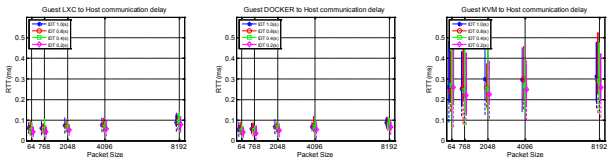


Figure 8: Round trip time between the host and LXC, Docker, and KVM guests.

4. MODELLING BBU PROCESSING TIME

The evaluation results in Section 3 confirm that uplink processing dominates the downlink, and that the total processing increases with PRB and MCS. However, the contribution of each underlying BBU functions to the total processing time and how they scale with the increase of PRB and MCS remains to be analysed so that an accurate model could be build. To this end, three main BBU functions that contribute the most to the total processing are considered including iFFT/FFT, (de)modulation, and (de)coding. For each function, the required processing time is measured on the Intel SandyBridge architecture with CPU frequency of 3.2GHz for different PRB, MCS, and virtualization environment (c.f. Figure 9).

The figures reveals that iFFT and FFT increase only with the PRB, while (de)modulation are (de)coding are increasing as a function of PRB and MCS. Each platform also adds a processing offset to each function. It can be seen that decoding and coding functions represent the most time consuming functions in uplink and downlink, and that the decoding is the dominant factor. Note that MCS 9, 16, and 27 corresponds to QPSK, 16QAM, and 64QAM with the highest coding rate. In OAI, decoding and encoding are based on the highly optimized SIMD integer DSP instructions (i.e. 64-bit MMX, 128-bit SSE2/3/4) used to speed up the processing. In a hypervisor-based virtualization, such instructions could add an extra delay if not supported by the hardware emulation layer (c.f. Figure 3).

From Figure 9, it can be observed that the uplink and downlink processing has two components: base processing and dynamic processing load. The base includes cell-processing (iFFT/FFT) for each PRB and the platform-specific processing relative to the reference GPP platform. The dynamic processing load includes user processing, i.e. (de)modulation and (de)coding, which is a linear function of allocated PRBs and MCS.[8] The reminder of user processing, namely scrambling, DCI coding, and PDCCH coding, is modelled as the root mean square error (RMSE) for each platform. Figure 10(a) shows the fitted curve for the total processing time for GPP and Figure 10(b) the RMSE for all platforms.

Based on the above results, a model is proposed to compute the total BBU uplink and downlink processing time for different PRB, MCS, and platform, and expressed by the following formula.

$$T_{\text{subframe}}(x,y,w)[\text{us}] = \underbrace{c[x]+p[w]}_{base\ processing} + \underbrace{u_r[x]}_{RMSE} + \underbrace{u_s(x,y)}_{dynamic\ processing}$$

where the triple (x, y, w) represents PRB, MCS, and platform. The $c[x]$ and $p[w]$ are the base offsets for the cell and platform

[8]Note that the dynamic processing load also depends on the SNR and the channel quality.

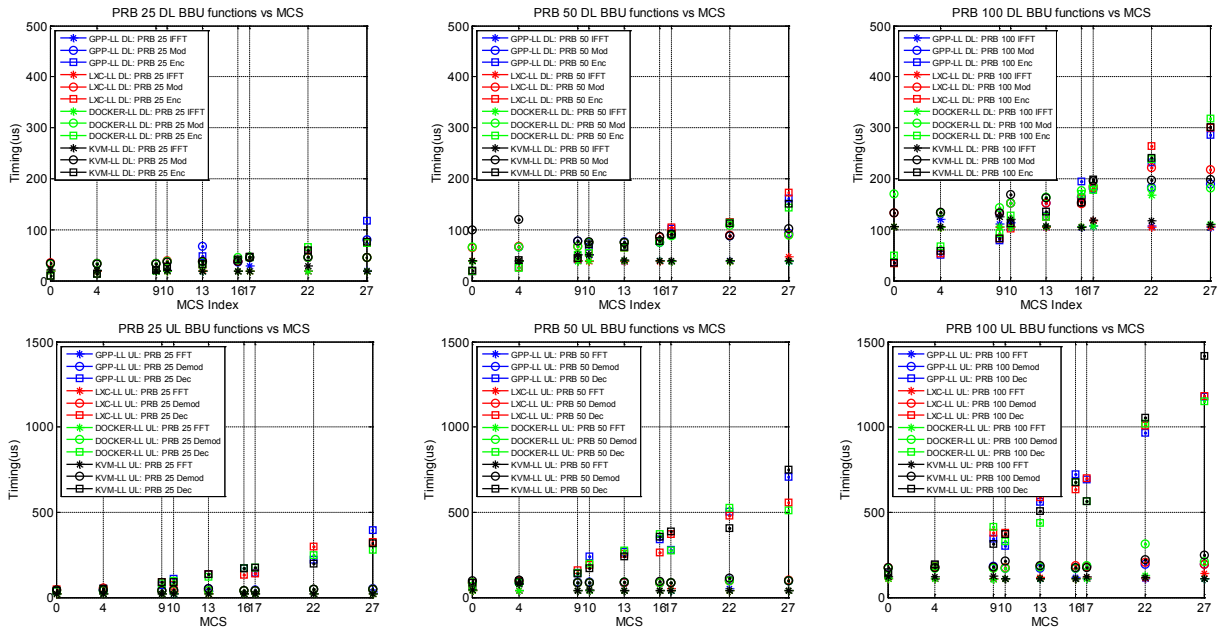
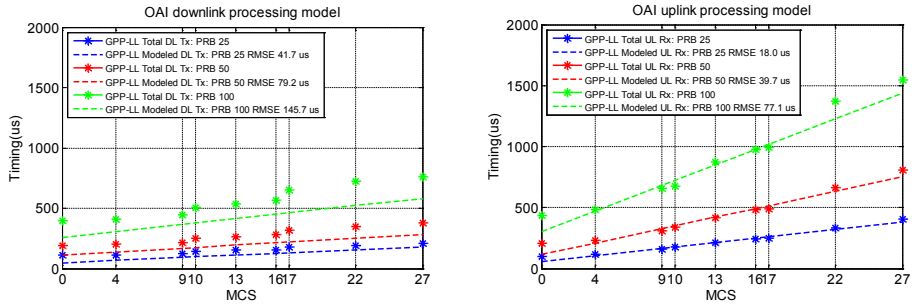
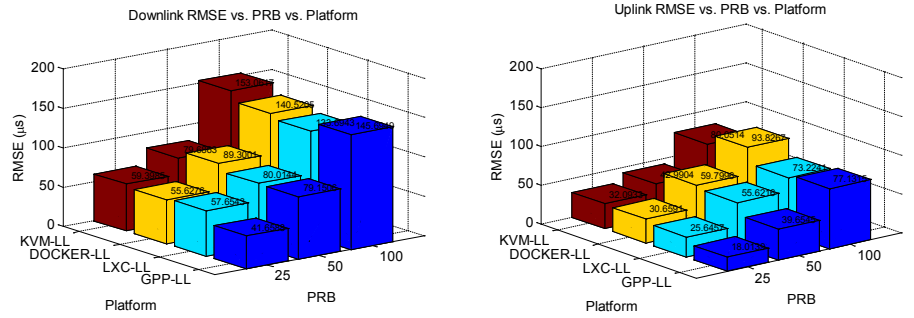


Figure 9: Contribution of (i)FFT, (de)modulation, and (de)coding to the total BBU processing for different PRB, MCS, and platforms.



(a) *Fitted curves for GPP-LL platform*



(b) *RMSE for all platforms*

Figure 10: Modeling BBU processing time.

processing, $u_r[x]$ is the reminder of user processing, and $u_s(x, y)$ is the specific user processing that depends on the allocated PRB and MCS. The $u_s(x, y)$ is linearly fitted to $a(x)y + b(x)$, where a, b are the coefficients, and y is the MCS. Table 2 and 3 provide the downlink and uplink model parameters of the equation 3 for an Intel-based SandyBridge architecture with the CPU frequency set to 3.2GHz. Note that the values have to be adjusted when targeting different BBU configuration (e.g. MIMO, Carrier aggregation), CPU architecture and frequency (c.f. Figure 4 and 5). For the considered setup, the accuracy of the model can be shown through an example. Let PRB to be 100, DL MCS 27, UL MCS 16, and platform LXC, the estimated total processing time is 723.5us (111.4+ 7.4+ 12*27+ 147+ 133.7) against 755.9us in downlink, and 1062.4us (108.8+ 13.2+ 41.9*16+ 196.8+ 73.2) against 984.9us in uplink.

Table 2: Downlink processing model parameters in us

x	c	p				$u_s(x,y)$		u_c			
		GPP	LCX	DOCKER	KVM	a	b	GPP	LCX	DOCKER	KVM
25	23.81	0	5.2	2.6	3.5	4.9	24.4	41.6	57.6	55.6	59.4
50	41.98	0	5.7	9.7	13	6.3	70	79.2	80	89.3	79.7
100	111.4	0	7.4	13	21.6	12	147	145.7	133.7	140.5	153

Table 3: Uplink processing model parameters in us

x	c	p				$u_s(x,y)$		u_c			
		GPP	LCX	DOCKER	KVM	a	b	GPP	LCX	DOCKER	KVM
25	20.3	0	5.4	4.8	8.8	11.9	39.6	18	25.6	30.6	32
50	40.1	0	6	9.2	15.8	23.5	75.7	39.6	55.6	59.8	42.9
100	108.8	0	13.2	31.6	26.6	41.9	196.8	77.1	73.2	93.8	80

5. DISCUSSIONS

This paper is an attempt to analyze three critical issues in processing radio access network functions in the cloud through modelling and measurements. The results reveal new directions to enable a cloud-native radio access network that are outlined below.

New functional split between BBU and RRH: To reduce the fornthaul data rate requirements, optimal functional split is required between BBU and RRH. In TX chain, full PHY layer can be moved from BBU to RRH (c.f. label 4 in Figure 1) in order to minimize the fronthaul capacity requirements as the operation of PHY layer remain deterministic as long as the L2/MAC layer provides transport blocks for all channels with the required pre-coding information. When it comes to RX chain, moving cell processing to RRH seems promising as it halves the fronthaul capacity requirements. Additional fronthaul capacity reduction can be obtained if part of user processing can be dynamically assigned to RRH (c.f. label 3 in Figure 1) depending on the number of UEs scheduled per resource elements and per RRH.

Number of CPU cores per BBU: In LTE-FDD, the total RX (UL) + TX (DL) processing should take less than 3 ms to comply with HARQ RTT, leaving 2 ms for RX and 1 ms for TX. Becasue TX requires the output of RX to proceed, the number of concurrent threads/cores per eNB subframe is limited to 3 even if each subframe is processed in parallel. By analyzing processing time for a 1ms LTE sub-frame, 2 cores at 3 GHz are needed to handle the total BBU processing of an eNB. One processor core for the receiver, assuming 16-QAM on the uplink, and approximately 1 core for the transmitter processing with 64-QAM on the downlink, are required to meet the HARQ deadlines for a fully loaded system. Processing load is mainly dominated by uplink and increases with growing PRBs and MCSs [10, 11]. Furthermore, the ratio and variation of downlink processing load to that of uplink also grows with the increase of PRB and MCS. With the AVX2/AVX3 optimizations, the computational efficiency is expected to double and thus a full software solution would fit with an average of 1 x86 core per eNB. Additional processing gain is achievable if certain time consuming functions are offloaed to a dedicated hardware accelerator.

Virtualization environment for BBU: When comparing results for different virtualization environments, the average processing times are very close making both container and hypervisor approach to RAN virtualization a feasible approach. However, the bare metal and LXC virtualization execution environments have slightly lower variations than that of DOCKER and KVM, especially with the increase of PRB and MCS increase. In addition, the I/O performance of container approach to virtualization proved to be very efficient. This suggests that fast packet processing (e.g. through DPDK) is required in hypervisor approach to minimize the packet switching time, especially for the fronthaul transport network. Due to the fact that containers are built upon modern kernel features such as `cgroups`, `namespace`, `chroot`, they share the host kernel and can benefit from the host scheduler, which is a key to meet real-time deadlines. This makes containers a cost-effective solution without compromising the performance.

6. POTENTIAL ARCHITECTURES

While the concept of C-RAN has been clearly defined, more research is needed to find an optimal architecture that maximizes the benefits behind C-RAN [7], and based on which a true proof-of-concept could be built [6].From the perspective of the operator such an architecture has to meet the scalability, reliability/resiliency, cost-effective requirements. However, from the perspective of the software radio application, two main requirements have to be met: (1) strict hard deadline to maintain the frame and subframe timing, and (2) efficient/elastic computational resources (e.g. CPU, memory) to perform intensive digital signal processing required for different transmission modes (beamforming, CoMP, and Massive MIMO).

Broadly, three main choices are possible to design a C-RAN, each of which provide a different cost, power, performance, and flexibility trade-offs.

- **Full GPP:** where all the processing (L1/L2/L3) is performed on the host/guest systems. According to China Mobile, the power consumption of the OpenAirInterface full GPP LTE softmodem is around 70w per carrier [6].
- **Accelerated:** where only certain functions, such as FFT/IFFT, are offloaded to a dedicated hardware such as an FPGA, GPU, and/or DSP, and the remaining functions operate on the host/guest OS. In this case, the power consumption can be reduced to around 13 18w per carrier.
- **System-on-Chip:** where the entire L1 is performed on a SoC and the reminder of the protocol stack runs on the host/guest OS. This can reduce the power consumption to around 8w per carrier.

As shown in Figure 11, the hardware platform can either be a full GPP or a hybrid. In the later case, all or part of the L1 functions might be offloaded to dedicated accelerators, which can be placed locally at the cloud infrastructure to meet the real-time deadline and provide a better power-performance trade-off or remotely at RRH to also reduce the data rate of fronthaul. It can be seen that a pool of base station (BS) is virtualized inside the same (or different) cloud infrastructure and shared among the cell sites. VBS can communicate with core networks (CN) through a dedicated interface (e.g. S1 in LTE), and with each other directly through another

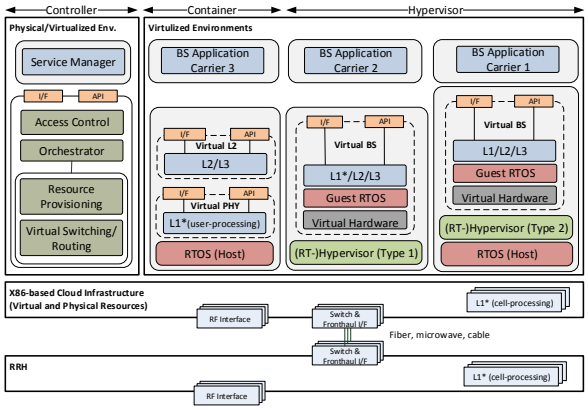


Figure 11: Potential C-RAN architectures.

interface (e.g. X2 in LTE). In addition, VBS can rely on the same cloud infrastructure to provide localized edge service such as content caching and positioning, and network APIs to interact with the access and core networks [15]. Different service compositions can be considered, ranging from all-in-one software radio application virtualization to per carrier, per layer or per function virtualization. The virtualization is performed either by a container engine or a hypervisor, under the control of a cloud OS, which is in charge of life-cycle management of a composite software radio application (orchestrator) and dynamic resource provisioning.

Nevertheless, a full GPP approach to RAN brings the cloud and virtualization even closer to the wireless world allowing to build a cloud-native RAN as a service along with the following principles [2, 12, 13]: [9]

- **Mircoservice Architecture and NFV:** breaks down the network into a set of horizontal functions that can be combined together, assigned with target performance parameters, mapped onto the infrastructure resources (physical or virtual), and finally delivered as a service. This implies that micro network functions are loosely coupled, reusable, composable, stateless, and discoverable.[10]
- **Scalability:** monitors the RAN events (e.g. workload variations, optimization, relocation, or upgrade) and automatically provision resources without any degradations in the required/agreed network performance (scale out/in).
- **Reliability:** shares the RAN contexts across multiple replicated RAN services to keep the required redundancy, and distributes the loads among them.
- **Placement:** optimizes the cost and/or performance by locating the RAN services at the specific geographic area subjected to performance, cost, and availability of the RF front-end and cloud resources.
- **Real-time Service:** offers a direct access to real-time radio information (e.g. radio status, statistics) for low-latency and high-bandwidth service deployed at the network edge [15].

[9]Two software implementations of a fully functional LTE/LTE-A already exist, namely by Amarisoft delivered as a commercial product and by OpenAirInterface delivered as an open-source software.

[10]Microservice architecture is in opposition to the so-called "monolithic" architecture where all functionality is offered by a single logical executable, see http://martinfowler.com/articles/microservices.html. It has to be noted that the micorservice architecture supports the ETSI NFV architecture [14], where each VNF can be seen as a service.

7. CONCLUSION

This paper investigates three critical issues towards the cloudification of the current LTE/LTE-A radio access network. Extensive set of experiments have been carried out to analyse the BBU processing load under different configurations and environments for Intel-based CPU architecture. The results have shown that the total processing scales up with PRB and MCS and that the uplink is the dominant processing load that might require to be splitted and/or accelerated. Coding and decoding functions represent the most time consuming BBU functions with high variability. It is found that container approach to virtualization provides slightly better performance than hypervisor approach.

Based on the results, a model is presented that accurately estimates the required uplink and down baseband processing as a function of PRB, MCS, and virtualization environment. The proposed model can be used for different optimization algorithms, namely (proactive) runtime resource prediction and allocation algorithms exploiting processing load variations to fewer resources in terms of computing, networking and storage for a given statistical realtime guarantee.

Acknowledgement

The research and development leading to these results has received funding from the European Framework Programme under FP7 grant agreement 318109 for MCN project and 318306 for NEWCOM# project. The author would also like to thank I. Alyafawi and Prof. R. Knopp for their useful discussions.

8. REFERENCES

[1] MITOLA, J. The software radio architecture. *IEEE Communication Magazine* (1995).

[2] WILDER, B. *Cloud Architecture Patterns*. O'Reilly, 2012.

[3] Mobile Cloud Networking project (FP7-ICT-318109). http://www.mobile-cloud-networking.eu.

[4] iJOIN: an FP7 STREP project co-funded by the European Commission under the ICT theme.

[5] LIN, Y., SHAO, L., ZHU, Z., WANG, Q., AND SABHIKHI, R. K. Wireless network cloud: Architecture and system requirements. *IBM Journal of Research and Development* (2010).

[6] CHINA MOBILE RESEARCH INSTITUTE. C-RAN The Road Towards Green RAN, 2013. White paper, v3.0.

[7] CHECKO, A., CHRISTIANSEN, H. L., YAN, Y., SCOLARI, L., KARDARAS, G., BERGER, M. S., AND DITTMANN, L. Cloud ran for mobile networks - a technology overview. *Communications Surveys Tutorials, IEEE* (2014).

[8] WUBBEN, D., ROST, P., BARTELT, J. S., LALAM, M., SAVIN, V., GORGOGLIONE, M., DEKORSY, A., AND FETTWEIS, G. Computing on 5g signal processing: Flexible centralization through cloud-ran. *IEEE Signal Processing Magazine* (2014).

[9] NGMN. Further Study on Critical C-RAN Technologies (v0.6). Tech. rep., The Next Generation Mobile Networks (NGMN) Alliance, 2013.

[10] BHAUMIK, S., ET AL. Cloudiq: A framework for processing base stations in a data center. In *Proc. ACM MobiCom* (2012).

[11] ALYAFAWI, I., SCHILLER, E., BRAUN, T., DIMITROVA, D., GOMES, A., AND NIKAEIN, N. Critical issues of centralized and cloudified lte fdd radio access networks. In *ICC* (2015).

[12] PATEL, M., JOUBERT, J., RAMOS, J. R., SPRECHER, N., ABETA, S., AND NEAL, A. Mobile-Edge Computing. Tech. rep., ETSI, white paper, 2014.

[13] NIKAEIN, N., KNOPP, R., GAUTHIER, L., SCHILLER, E., BRAUN, T., PICHON, D., BONNET, C., KALTENBERGER, F., AND NUSSBAUM, D. Demo – Closer to Cloud-RAN: RAN as a Service. *Proceedings of ACM MOBICOM Demonstrations* (2015).

[14] MCN D2.5. Final Overall Architecture Definition. Tech. rep., 2015.

[15] ETSI. Network Functions Virtualisation (NFV), White paper. Tech. rep., 2014.