Temporal isolation among LTE/5G network functions by real-time scheduling*

Tommaso Cucinotta[1], Mauro Marinoni[1], Alessandra Melani[1], Andrea Parri[1] and Carlo Vitucci[2]
[1]*Scuola Superiore Sant'Anna, Pisa, Italy*
[2]*Ericsson, Stockholm, Sweden*
{*name.surname*}*@sssup.it, carlo.vitucci@ericsson.com*

Keywords: Real-Time Scheduling; Virtual Radio Access Network; Temporal Isolation; Quality of Service Control

Abstract: Radio access networks for future LTE/5G scenarios need to be designed so as to satisfy increasingly stringent requirements in terms of overall capacity, individual user performance, flexibility and power efficiency. This is triggering a major shift in the Telcom industry from statically sized, physically provisioned network appliances towards the use of virtualized network functions that can be elastically deployed within a flexible private cloud of network operators. However, a major issue in delivering strong QoS levels is the one to keep in check the temporal interferences among co-located services, as they compete in accessing shared physical resources.
In this paper, this problem is tackled by proposing a solution making use of a *real-time scheduler* with strong temporal isolation guarantees at the OS/kernel level. This allows for the development of a *mathematical model* linking major parameters of the system configuration and input traffic characterization with the achieved performance and response-time probabilistic distribution. The model is verified through extensive experiments made on Linux on a synthetic benchmark tuned according to data from a real LTE packet processing scenario.

1 INTRODUCTION

Radio access networks (RANs) are undergoing a steep evolution towards higher overall capacity and coverage across the territory and enhanced performance for individual mobile devices, both in terms of throughput and round-trip latency towards cloud computing services (López-Pérez et al., 2015; Clo, 2015; Cis, 2016; Fiorani et al., 2015). This, coupled with an increasing need for power-efficient and adaptive solutions supporting highly variable traffic conditions, results in a strong demand for flexible network architectures, designed applying principles from distributed, cloud and service-oriented computing, thanks also to the progressive convergence towards IP-based networks, as witnessed by LTE.

Traditional deployments suffer of a great underutilization of resources throughout the majority of (off-peak) operations, leading to drawbacks in terms of energy consumption, as well as unsustainable OPEX. As opposed to this, the novel paradigm of Network Function Virtualization (NFV) (NFV Industry Specif. Group, 2012) is spreading, where IP-based network functions are virtualized into software components which are deployed, consolidated and scaled across a flexible infrastructure, managed according to a private cloud deployment model (Armbrust et al., 2009; Hawilo et al., 2014).

Design of Telecom applications needs traditionally to focus not only on functional correctness, but also on non-functional properties such as availability and performance. The latter ones have been facilitated by classical designs based on having separate physical appliances dedicated to different network functions. However, the efficiency of new NFV-based designs relies on aggressive consolidation strategies where various possibly heterogeneous functions (e.g., GSM, LTE, IMS, security-related, billing, monitoring) can be hosted on the same system, leading to increased temporal interferences among them due to competing accesses to shared physical resources.

Therefore, one of the key success factors in new designs is the degree at which the various co-located functions can actually be temporally isolated from each other. The traditional way to control the reciprocal interference among co-deployed software components is by *partitioning* the available physical resources. For example, a network function can be de-

*This work was partially founded by Ericsson. The authors want to thank Freescale for their support with the reference platform.

ployed as a *virtual machine* or *container*[2] restricted to operate onto a subset of the cores actually available on the physical server, with a reduced overall amount of RAM, local persistent storage, and even restricted networking capabilities (throttling). However, while memory partitioning and network throttling can be done with quite a degree of flexibility, partitioning of physical cores constitutes a major restriction on the possible available choices, and the number of functions that can be deployed onto the same system. In many cases, to enhance resources utilization, physical cores are shared among multiple virtualized functions, but there are no ways to control exactly how much time the shared processors are dedicated to each such function, resulting in services suffering of very variable and unpredictable performance fluctuations.

Contributions. In this paper, the problem of temporal isolation among a number of network functions co-located on the same physical node is tackled. A design is proposed for a multi-core server capable of hosting multiple functions with predictable performance levels, leveraging two main building blocks: a) use of a *reservation-based real-time scheduler at the OS kernel level*, providing scheduling guarantees to, and temporal isolation across, individual functions; b) an insightful *queueing-based mathematical model* that links performance statistics that can be expected from the system with its configuration options in terms of scheduling parameters and input traffic characterization. The CPU scheduler in use allows for guaranteeing each deployed function both the computational bandwidth and the precise time granularity over which CPU cycles are granted. The model is verified through experimental validation made on a synthetic workload and scenario inspired to real data coming from an LTE use-case.

2 PROBLEM PRESENTATION

2.1 NFV and virtualized RAN

In the context of radio access networks (RANs), the NFV paradigm is translating into a number of solutions based on the concept of virtualizing RAN functions (Clo, 2015; Haberland et al., 2013; Costa-Perez et al., 2013; Maaref et al., 2014; Cis, 2016). In these architectures, packet processing functions are moved away from the antenna equipment and consolidated into geo-distributed private-cloud data centers. For example, in the context of LTE (see Figure 1), there have been proposals to move both the physical (L1) and the MAC (L2) layers to a shared data center, letting antennas become merely *remote radio heads* connected to a centralized Remote Equipment Controller (REC) in the private cloud via high-performance connectivity such as CPRI (CPR, 2015). However, such an approach poses very high bandwidth requirements and stringent-to-prohibitive round-trip latency constraints (in the order of $100\mu s$) on the CPRI link. A certainly more viable approach is the one to centralize in the NFV infrastructure only the MAC layer processing, where the requirement on round-trip latency becomes way more satisfiable, e.g., air interface L2 packets need to be processed in less than $6ms$ (Cis, 2016), where 3.66ms is a typically acceptable L2 round-trip latency (Pichon, 2014).

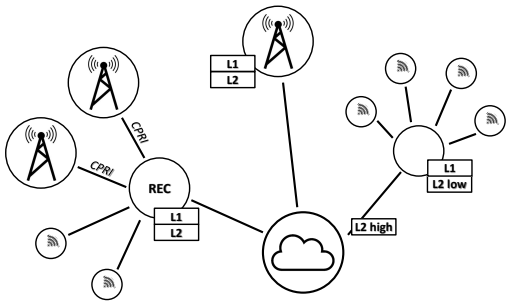

Figure 1: Possible RAN/VRAN deployment models.

Also, L2 processing functions are usually CPU-intensive, e.g., as they usually involve encryption/decryption of data. Therefore, traditional approaches to fulfilling the tight latency/processing constraints resort to dedicated hardware accelerators in such cases. However, novel cloud-based paradigms usually rely on commodity hardware so it makes sense to focus on software-only solutions.

2.2 Problem statement

Consider the scenario in Figure 2: several network functions are deployed onto a single Digital Unit, performing L2 packet processing functions for LTE cellular networks. We focus on controlling the temporal interference as due to co-scheduling of CPU-intensive network functions on the same cores and modeling their resulting performance. Therefore, interferences

[2]In what follows, we refer to the term virtualized function and virtual machine in a loose sense, including the possibility that these functions are deployed as virtual machines on a hypervisor, or simply deployed within different containers or isolated processes within the same operating system or runtime environment.

due to competing access to other resources than the CPU, like network or disk or cache/memory hierarchy, is post-poned to future extensions of the work.

Traffic coming from each cell is handled by one or multiple threads that need to process each received packet within a response-time R^*, at least for a (high) percentage of the traffic ϕ. In other words, denoting with R the stochastic variable representing the response-time of each of our L2 traffic processing functions, the probabilistic Service-Level Objective (SLO) under consideration is formally stated as:

$$\Pr\{R \leq R^*\} \geq \phi. \quad (1)$$

The considered reference system is composed of a single server with m identical cores, where multiple worker threads are deployed, serving traffic from a number of cells. The considered scenario and analysis shown in Section 4 deals explicitly with the case of n workers serving traffic from the same cell, with a random load-balancing of the traffic among them. Input traffic from each cell is assumed to be a Poisson process, with independent identically and exponentially distributed (i.i.d.) inter-arrival times, with average aggregate arrival rate λ_G and average inter-arrival times of $\frac{1}{\lambda_G}$. Processing times for workers are assumed exponentially distributed and i.i.d. as well, with average service rate μ and average computation time $\frac{1}{\mu}$, when processed in isolation on a single core. Note that it is assumed that the platform does not have CPU frequency switching capabilities, or that they are disabled if present, otherwise the input traffic volume would influence the packet processing times. Indeed, in the majority of modern architectures, static power consumption is dominant w.r.t. dynamic one. Microprocessors based on such technologies save considerable more energy when in deep-sleep states, so it is convenient to run them at the maximum frequency so that the amount of time they spend in the sleep state is maximized (Benini et al., 2000).

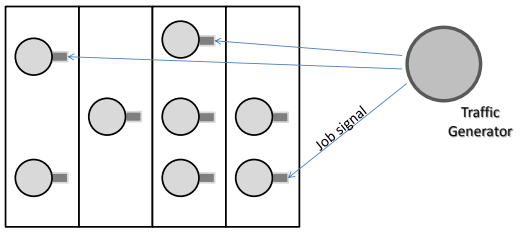

Figure 2: Reference scenario: a cell traffic is processed by one or more workers deployed on a multi-core server.

In this paper, a model is shown that, given a characterization of the input process, allows for tuning the scheduling configuration for the workers so as to meet the SLO requirements of Equation (1). As it will be clear shortly, the presented closed-form model provides also additional insights to system designers.

3 RESERVATION-BASED CPU SCHEDULING

Resource reservation scheduling (Mercer et al., 1993) is a well-known model for sharing a physical CPU/core among a number of tasks, in a way that each one of them can be granted a pre-fixed amount of time Q on the CPU (called *budget*) every allocation window of duration P (called *scheduling period*), providing *temporal isolation* among co-located tasks. A specific algorithm implementing this paradigm is the *Constant Bandwidth Server* (CBS) (Abeni and Buttazzo, 1998), a technique that gained attention from the Linux community recently as due to its availability within the mainline kernel since version 3.14 (Lelli et al., 2016), in the form of the SCHED_DEADLINE scheduling policy. [3] Under specific conditions (see Section 3), this allocation of the CPU is granted by the scheduler, independently of how other tasks behave in the system, effectively providing temporal isolation among them. On a multi-core systems, SCHED_DEADLINE can provide both global and partitioned EDF (G-EDF and P-EDF, respectively).

SCHED_DEADLINE adds to the real-time scheduling features of the Linux kernel in serving soft real-time workloads, making it an increasingly appealing platform for the execution of virtualized network functions with predictable performance levels (Vitucci et al., 2014; Clo, 2015).

Schedulability guarantees. SCHED_DEADLINE is known to provide temporal isolation among worker tasks with no deadline misses if either of the following conditions is satisfied: 1) if tasks are globally scheduled, reservations respect the G-EDF schedulability bound (Goossens et al., 2003) for m cores,

$$\sum_{i=1}^{n} \frac{Q_i}{P_i} \leq m(1-U_{max})+U_{max}, \quad (2)$$

where $U_{max} \triangleq \max_{i=1,\ldots,n}\left\{\frac{Q_i}{P_i}\right\}$; if $Q_i/P_i = U\ \forall i$:

$$U \leq \frac{m}{n-1+m}; \quad (3)$$

if $n <= m$ G-EDF will migrate each task to a different core with the only constraint that $Q_i/P_i \leq 1$ for each i; 2) if tasks are partitioned among cores, reservations need to respect the P-EDF (single-processor)

[3] https://www.kernel.org/doc/Documentation/scheduler/sched-deadline.txt

schedulability bound (Liu and Layland, 1973): assuming each core j hosts n_j workers with $\sum_{j=1}^{m} n_j = n$:

$$\sum_{i=1}^{n_j} \frac{Q_{j,i}}{P_{j,i}} \leq 1, \quad \text{for each } j = 1, \ldots, m. \tag{4}$$

The traditional way to enforce timing constraints and full isolation assigns $\{Q_i\}$ equal to the task worst-case computing time and $\{P_i\}$ equal to the minimum inter-arrival times for subsequent requests. However, such an approach is aimed at *hard* real-time systems, where even a single deadline miss cannot be tolerated.

In this paper, a *soft* real-time approach is undertaken instead, where the system has to satisfy precise timing requirements expressed as a probabilistic SLO. Therefore, it is possible to trade exact schedulability for efficiency, operating the system at a point that is beyond the above teheoretical schedulability bound, especially for the G-EDF case, because the test in Equation (2) is quite pessimistic (see Section 5).

4 APPROACH AND ANALYSIS

In this section, a probabilistic response-time analysis based on queueing theory is presented, for the LTE/5G application scenario described in Section 2.

4.1 SCHED_DEADLINE queueing model

In the proposed architecture, network functions are deployed by scheduling them using SCHED_DEADLINE, with scheduling parameters $\{(Q_i, P_i)\}_{i=1}^{n}$ to be decided as detailed below. Thanks to the temporal isolation property enforced by SCHED_DEADLINE (see Section 3), each worker task i with a SCHED_DEADLINE runtime of Q_i and period of $P_i \geq Q_i$ can be seen as an isolated queueing system whose service rate μ_i has a slow-down factor of $\frac{Q_i}{P_i}$ w.r.t. the maximum sequential service rate μ:

$$\mu_i = \frac{Q_i}{P_i}\mu \tag{5}$$

If Poissonian traffic from a cell is randomly spread across more workers so that a percentage p_i of the aggregate traffic λ_G is picked up by worker i, then sub-flows in input to the workers are still Poissonian with an average rate of $\lambda_i = p_i \cdot \lambda_G$. Since both arrival and service rates are assumed to be exponentially distributed, the behavior of each worker can be approximated as an isolated $M/M/1$ queue (Kendall, 1953). Refer to Section 5 for an experimental validation of the approximation and the derived results.

General $M/M/1$ results (Gross et al., 2008) are specialized to our case in what follows, where for simplicity the discussion is referred to a single cell whose entire aggregate traffic λ_G is split across the n workers to be deployed onto an m-cores platform.

Let ρ be defined as $\rho \triangleq \frac{\lambda_i}{\mu_i} \equiv \frac{\lambda_G P}{nQ\mu}$ and assumed constant across all workers (to ease notation, the i index is omitted, when the discussion is implicitly referring to a specific worker or all workers have the same configuration). First, let us consider the schedulability condition for the scheduler as detailed in Section 3. Under P-EDF, assuming a system filled up with reserved tasks, the maximum theoretical computational bandwidth is derived from Equation (4):

$$\frac{Q}{P} = \frac{m}{n} \implies \rho = \frac{1}{m}\frac{\lambda_G}{\mu}. \tag{6}$$

Under G-EDF, instead, Equation (2) leads to:

$$\frac{Q}{P} = \frac{m}{n-1+m} \implies \rho = \frac{\lambda_G}{\mu}\frac{n-1+m}{nm}. \tag{7}$$

In addition to the theoretical schedulability conditions, it is useful to make explicit the general *stability* condition for the system, e.g., that incoming traffic on each queueing system on average does not exceed the server processing capability:

$$\rho < 1 \implies \frac{Q}{P} > \frac{\lambda_G}{n\mu}. \tag{8}$$

Once the servers are properly isolated, each of them behaves as an independent M/M/1 queue, thus the average response time R_i and waiting time W_i are obtained as:

$$E[R_i] = E[W_i] + \frac{1}{\mu_i} = \frac{1}{\mu_i - \lambda_i}. \tag{9}$$

The response-time R_i distribution is known to be exponential as well with parameter $\mu_i - \lambda_i$, thus we get:

- R_i probability distribution function (PDF):

$$f_{R_i}(t) = (\mu_i - \lambda_i)\, e^{-(\mu_i - \lambda_i)t} \text{ for each } t > 0, \tag{10}$$

- R_i cumulative distribution function (CDF):

$$F_{R_i}(t) = 1 - e^{-(\mu_i - \lambda_i)t} \quad \text{for each } t > 0. \tag{11}$$

4.2 Choice of the (Q_i, P_i) parameters

The formal SLO requirement for the system, as given in Equation (1), can now be described as: "ϕ^{th} *percentile of the response-time* $R \leq$ *a given threshold* R^*". This requirement can be formalized as follows:

$$F_R(R^*) \quad \geq \quad \phi, \tag{12}$$

where F_R is the response-time CDF introduced in Section 4. In the case of an $M/M/1$ system with parameters μ_i and λ_i, Equation (12) becomes:

$$\frac{Q_i}{P_i} \geq \frac{1}{\mu}\left[\lambda_i - \frac{ln(1-\phi)}{R^*}\right]. \quad (13)$$

Note that any (Q_i, P_i) pair satisfying the probabilistic SLO in Equation (13), satisfies also the M/M/1 stability condition in Equation (8). On the other hand, schedulability of the reservations on the server has to be ensured separately, e.g., according to Equations (7) or (6) if G-EDF or P-EDF is used, respectively.

From Equation (13), the full specification of Q and P is obtained observing that it is convenient to set $P = R^*$, to avoid temporal interferences and not cause excessive scheduling on the platform. The point is further detailed in Section 5.2.

4.3 Maximum sustainable arrival rate

The $M/M/1$ model described in Section 4 allows us to compute the maximum sustainable aggregate arrival rate for a given platform configuration: from Equation (12), we have

$$\lambda_G = n\lambda_i \leq n\left[\frac{Q}{P}\mu + \frac{\ln(1-\phi)}{R^*}\right], \quad (14)$$

where the logarithm brings a negative contribution.

We remark that the obtained upper-bound to λ_G permits achieving the target response-time percentile, but it does not guarantee the full temporal isolation of the reservation servers. Temporal isolation can be guaranteed by considering the maximum computational bandwidth $\frac{Q}{P}$ saturating the system under P-EDF in Equation (6), that is:

$$\lambda_G \leq m\mu + \frac{n}{R^*}\ln(1-\phi). \quad (15)$$

On the other hand, using the maximum computational bandwidth as from theoretical G-EDF schedulability in Equation (7), we obtain:

$$\lambda_G \leq \frac{nm}{n-1+m}\mu + \frac{n}{R^*}\ln(1-\phi). \quad (16)$$

5 EXPERIMENTAL VALIDATION

In this section, experimental results are presented that validate the analysis in Section 4. All experiments have been conducted on a Freescale LS2085A RDB (ARMv8) board, with 8 cores running at 1.8GHz and 16GB of RAM, a promising platform for energy-efficient data-plane processing and NFV. The system boots an image with Linux 4.1.8, built using Yocto [4]. A synthetic application was built around the LTE scenario described in Section 2: a client acting as traffic generator is capable of submitting requests according to tunable exponential inter-arrival times; a multi-threaded server is able to serve requests received from the client, taking a tunable exponentially distributed computation time; these are designed to dump on disk the full set of observed response-times throughout each experiment. The server can configure its threads using tunable SCHED_DEADLINE settings, including the use of either G-EDF or P-EDF.

[4] https://www.yoctoproject.org/

5.1 Scenario parameters

In the considered scenario, $n = 8$ worker threads have been instantiated, scheduled under G-EDF across $m = 4$ out of the available cores, and configured to serve traffic with a packet processing rate of $\mu \cong 5300 pkt/s$. A single client generator submitted requests at an average rate of $\lambda_G = 15000 pkt/s$, obtaining an individual (worker) traffic average rate of: $\lambda_i = \frac{\lambda_G}{n} = 1875 pkt/s$. The cell traffic type and rate, as well as the workers service rate, have been chosen by best-fitting parameters over data gathered from a real LTE installation. Also, the same industrial use-case mandated the needed probabilistic SLO parameters: 99% of the packets served within $2ms$, namely: $R^* = 2ms$ and $\phi = 0.99$. The presented results have been obtained via 1-minute long runs, each relying on $15000 * 60 = 900K$ measured response times.

5.2 Experimental results

Theoretical results from Section 4, specifically Equations (13) and (8), can easily be applied for the just explicited configuration, obtaining, among others:

- min Q/P for stability $\frac{Q}{P} > \frac{\lambda_G}{n\mu} \cong 35.38\%$;
- min Q/P for SLO: $\frac{Q}{P} \geq \frac{1}{\mu}\left[\lambda_i - \frac{ln(1-\phi)}{R^*}\right] \cong 78.8\%$.

These results have been compared with the obtained experimental data, shown in Figure 3, where the obtained response-time 99^{th} percentile (on the Y axis) is plotted with various Q/P ratios (on the X axis), and various scheduling periods (different data sets). The probabilistic SLO requirement is clearly satisfied when $\frac{Q}{P} \geq 0.7$, as per the theoretical expectation.

Selection of the period P_i. Equation (13) provides a choice for the Q/P ratio, for a worker that needs to service traffic with precise probabilistic guarantees. The exact parameters can be determined based on the following considerations, focusing on the choice of the period P. First, a smaller period leads to increased scheduling overheads and context switches

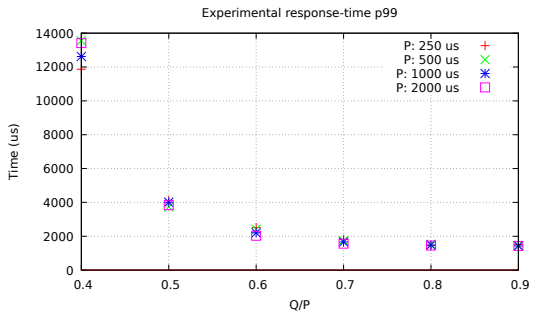

Figure 3: Experimental thresholds (R^*) when $\phi = 0.99$, for different values of the ratio Q_i/P_i.

(roughly 2 per scheduling period per task); indeed, the obtained experimental 90^{th} and 99^{th} percentiles of the response times were slightly better (lower) with higher periods, among the ones tried (see Figure 3). Second, the scheduling period determines the time granularity based on which SCHED_DEADLINE schedules each reserved task, thus the scheduling delay is roughly bounded by P; indeed, experimental results show slightly worse worst-case response-times when using higher periods. As also found experimentally, setting $P = 2ms$ allows for meeting the temporal requirements without excessive overheads.

In another 1-min experiment, the context switches in the platform throughout the whole run have been counted, in order to highlight how many additional context switches can happen with too small of a period. Results are reported in Figure 4. With a period of $2ms$, SCHED_DEADLINE achieves a reduction in the number of context switches of about 10% compared to SCHED_FIFO. However, using arbitrarily small periods can lead SCHED_DEADLINE to overwhelm the platform with an excess of context switches (e.g., +40% extra context switches with a $100\mu s$ period).

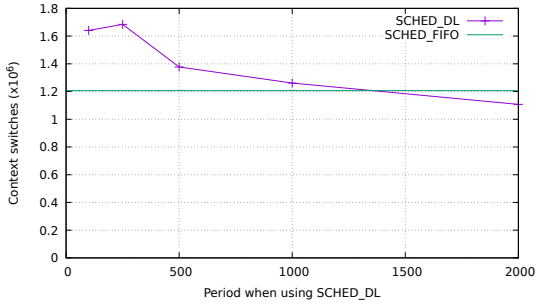

Figure 4: Number of context switch for SCHED_DEADLINE and SCHED_FIFO, when varying the reservation periods.

Temporal isolation. The advantage of using SCHED_DEADLINE resides in its capability to provide temporal isolation with a specific time granularity among workers. In this section, this is highlighted through an experiment comparing SCHED_DEADLINE vs SCHED_FIFO in presence of an anomalous traffic overloading one of the workers. The overload has been simulated by using a higher probability of 0.475 for one of the workers and a lower probability of 0.075 for the remaining 7 workers. Then, raising the overall traffic rate to $\lambda_G = 25000 pkt/s$, we obtained 7 individual rates at the usual $1875 pkt/s$ while the 8th individual rate is at the higher value of $11875 pkt/s$. Budgets and periods for SCHED_DEADLINE have been set to $900\mu s$ and $1000\mu s$, respectively.

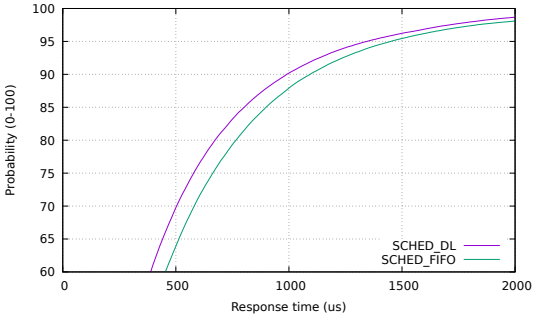

Figure 5: Response time CDFs in the presence of overload, obtained under the scenario of Section 5.2.

Figure 5 compares the response-time CDF obtained for one of the regular workers when using SCHED_DEADLINE and when using SCHED_FIFO. The response times are higher under SCHED_FIFO, due to the interference of the overloaded worker, exhibiting a $+300us$ increase in the 99th percentile of the response times compared to the ones obtained under SCHED_DEADLINE. On the other hand, the overloaded worker managed to serve 26% more packets under SCHED_FIFO, because its computing capabilities were not restricted by the scheduling budget as in the case of SCHED_DEADLINE, as expected.

6 RELATED WORK

In the context of the Long Term Evolution (LTE) technology, virtualization-based mobile network architectures are now widely accepted, and NFV is now recognized as an established solution to be deployed in core networks and data centers (Zaki et al., 2011; Sama et al., 2015; Nguyen et al., 2016). However, scarce attention has been dedicated so far to the problem of QoS assurance in edge nodes. This paper tries to fill this gap by addressing predictable execution

within edge nodes to ensure QoS control and temporal isolation, by resorting to an EDF-based real-time scheduling strategy. On a related note, in (Jansen and Mullender, 2003) authors propose to apply EDF-based real-time scheduling within a Plan9-based experimental base-station software stack, albeit the focus of the work is on preventing priority inversion through a specific deadline inheritance mechanism. A similar approach has been more recently presented in (Vitucci et al., 2014), but no attempt has been made to build a precise model of the function that takes properly into consideration the scheduling parameters, as done in this paper.

The problem of resource partitioning for temporal isolation has been largely investigated in the real-time scheduling literature. The main concern in this regard was to ensure that multiple software components could be deployed on the same physical cores ensuring that any task overrun could not affect the execution of other tasks and disrupt system integrity. This need mainly arises whenever a timely service has to be ensured in a system with heterogeneous timing requirements or dynamic computational workload. To achieve this property, several different Resource Reservation algorithms have been developed. The earliest examples (Rajkumar et al., 1998; Abeni and Buttazzo, 1998) have been developed in the context of multimedia applications, with the intent to offer predictable levels of quality-of-service (QoS) to real-time workload in highly heterogeneous environments. Resource reservation techniques, which had originally been proposed for the execution of independent tasks on single-core systems, were also extended to cope with hierarchical scheduling systems (Feng and Mok, 2002) and multi-core platforms (Shin et al., 2008; Easwaran and Andersson, 2009).

The effectiveness of these scheduling mechanisms is also supported by the recent introduction of the SCHED_DEADLINE scheduling class in the Linux kernel (Lelli et al., 2016). Traditional Linux scheduling classes either do not provide real-time guarantees (see SCHED_OTHER policy, which schedules activities in a best-effort way), or implement priority-based real-time scheduling with no resource reservation capabilities (see SCHED_FIFO or SCHED_RR policies). Vice versa, SCHED_DEADLINE allows combining traditional multi-core scheduling with resource reservations using a variant of the Constant Bandwidth Server (CBS) algorithm (Abeni and Buttazzo, 1998) to obtain temporal isolation among tasks. Interestingly, SCHED_DEADLINE does not require any particular assumption on tasks characteristics, allowing it to serve highly heterogeneous workloads.

A remarkable work in this area has been carried out in the context of the IRMOS European project [5], where an implementation of the EDF/CBS algorithm within the Linux kernel has been used to provide strong scheduling guarantees to KVM-based VMs running in a virtualized real-time cloud infrastructure. However, one of the critical issues that emerged was the one of tuning proper values for the scheduling parameters, and how to link them to the overall performance of the deployed application. In this work, we mark a significant step forward in such an issue, developing a closed-form mathematical model capable of capturing the relationship between system configuration, input traffic characterization and expected performance of the virtualized network functions.

7 CONCLUSIONS AND FUTURE WORK

In this paper, an approach has been proposed to provide temporal guarantees to LTE traffic processing functions, based on the SCHED_DEADLINE Linux scheduler. This allows for modelling the performance of individual functions applying classical queueing theory. Experimental data shows that the proposed model matches the application behavior. SCHED_DEADLINE allows for controlling the computational bandwidth ratio dedicated to each function, as well as the time granularity of the schedule.

The presented model is shown to provide easy-to-use closed-form formulas under Poissonian traffic assumptions, providing insightful relationships among parameters related to the system and scheduler configuration, and expected performance figures.

The presented work has various limitations that need to be addressed in additional future research. First, the proposed methodology needs to be applied in a more realistic scenario, where multiple cells submit traffic to functions deployed throughout a distributed NFV infrastructure. Second, the paper focused on CPU-dominated workloads, but in real life, there will also be a mix of data-intensive activities that saturate disk access or available NIC bandwidth, before the CPU. The modeling and analysis technique needs to be extended to cope with these more involved scenarios. Furthermore, it would be interesting to verify what integration issues arise for integrating the reservation scheduler paradigm within a real NFV infrastructure. This might be done, e.g., extending OPNFV[6] or other frameworks.

[5] http://http://www.irmosproject.eu/

[6] https://www.opnfv.org/

REFERENCES

(2015). Cloud RAN – Ericsson White Paper. www.ericsson.com/res/docs/whitepapers/wp-cloud-ran.pdf.

(2015). Common Public Radio Interface (CPRI) v7.0 – Interface Specification. http://www.cpri.info/downloads/CPRI_v_7_0_2015-10-09.pdf.

(2016). Cisco 5G Vision Series: Small Cell Evolution. Cisco Whitepaper.

Abeni, L. and Buttazzo, G. (1998). Integrating multimedia applications in hard real-time systems. In *Proc. IEEE Real-Time Systems Symposium*, pages 4–13, Madrid, Spain.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R. H., Konwinski, A., Lee, G., Patterson, D. A., Rabkin, A., Stoica, I., and Zaharia, M. (2009). Above the clouds: A berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley.

Benini, L., Bogliolo, A., and Micheli, G. D. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 8(3):299–316.

Costa-Perez, X., Swetina, J., Guo, T., Mahindra, R., and Rangarajan, S. (2013). Radio access network virtualization for future mobile carrier networks. *IEEE Communications Magazine*, 51(7):27–35.

Easwaran, A. and Andersson, B. (December 2009). Resource sharing in global fixed-priority preemptive multiprocessor scheduling. In *Proceedings of the 30th IEEE Real-Time Systems Symposium (RTSS)*, Washington, DC, USA.

Feng, X. and Mok, A. K. (December 2002). A model of hierarchical real-time virtual resources. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium (RTSS)*, Austin, TX, USA.

Fiorani, M., Skubic, B., Mårtensson, J., Valcarenghi, L., Castoldi, P., Wosinska, L., and Monti, P. (2015). On the design of 5G transport networks. *Photonic Network Communications*, 30(3):403–415.

Goossens, J., Funk, S., and Baruah, S. (2003). Priority-driven scheduling of periodic task systems on multiprocessors. *Real-Time Systems*, 25(2):187–205.

Gross, D., Shortle, J. F., Thompson, J. M., and Harris, C. M. (2008). *Fundamentals of Queueing Theory*. Wiley-Interscience, 4th edition.

Haberland, B., Derakhshan, F., Grob-Lipski, H., Klotsche, R., Rehm, W., Schefczik, P., and Soellner, M. (2013). Radio base stations in the cloud. *Bell Labs Technical Journal*, 18(1):129–152.

Hawilo, H., Shami, A., Mirahmadi, M., and Asal, R. (2014). NFV: state of the art, challenges, and implementation in next generation mobile networks (vepc). *IEEE Network*, 28(6):18–26.

Jansen, P. G. and Mullender, S. (2003). Real-time in plan 9 : a short overview.

Kendall, D. G. (1953). Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded markov chain. *The Annals of Mathematical Statistics*, 24(3):338–354.

Lelli, J., Scordino, C., Abeni, L., and Faggioli, D. (2016). Deadline scheduling in the linux kernel. *Software: Practice and Experience*, 46(6):821–839.

Liu, C. L. and Layland, J. W. (1973). Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journel of the ACM*, 20(1):46–61.

López-Pérez, D., Ding, M., Claussen, H., and Jafari, A. H. (2015). Towards 1 gbps/ue in cellular systems: Understanding ultra-dense small cell deployments. *IEEE Communications Surveys Tutorials*, 17(4):2078–2101.

Maaref, A., Ma, J., Salem, M., Baligh, H., and Zarin, K. (2014). Device-centric radio access virtualization for 5g networks. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 887–893.

Mercer, C. W., Savage, S., and Tokuda, H. (1993). Processor Capacity Reserves: An Abstraction for Managing Processor Usage. In *Proc. 4th Workshop on Workstation Operating Systems*.

NFV Industry Specif. Group (2012). Network Functions Virtualisation. Introductory White Paper.

Nguyen, V.-G., Do, T.-X., and Kim, Y. (2016). Sdn and virtualization-based lte mobile network architectures: A comprehensive survey. *Wireless Personal Communications*, 86(3):1401–1438.

Pichon, D. (2014). Mobile Cloud Networking FP7 European Project: Radio Access Network as a Service. 4th Workshop on Mobile Cloud Networking. Available at: http://docplayer.net/3714038-Mobile-cloud-networking-fp7-european-project-radio-access-network-as-a-service.html.

Rajkumar, R., Juvva, K., Molano, A., and Oikawa, S. (January 1998). Resource kernels: A resource-centric approach to real-time and multimedia systems. In *SPIE/ACM Conference on Multimedia Computing and Networking*, San Jose, CA, USA.

Sama, M. R., Contreras, L. M., Kaippallimalil, J., Akiyoshi, I., Qian, H., and Ni, H. (2015). Software-defined control of the virtualized mobile packet core. *IEEE Communications Magazine*, 53(2):107–115.

Shin, I., Easwaran, A., and Lee, I. (July 2008). Hierarchical scheduling framework for virtual clustering of multiprocessors. In *20th Euromicro Conference on Real-Time Systems (ECRTS)*, Prague, Czech Republic.

Vitucci, C., Lelli, J., Parri, A., and Marinoni, M. (2014). A Linux-based Virtualized Solution Providing Computing Quality of Service to SDN-NFV Telecommunication Applications. In *Proceedings of the 16th Real Time Linux Workshop (RTLWS 2014)*, pages 12–13, Dusseldorf, Germany.

Zaki, Y., Weerawardane, T., Gorg, C., and Timm-Giel, A. (May 2011). Multi-qos-aware fair scheduling for LTE. In *73rd IEEE Vehicular technology conference (VTC-Spring)*, Budapest, Hungary.