# Performance analysis of resource pooling for network function virtualization

Veronica Karina Quintuna Rodriguez, Fabrice Guillemin

# Performance Analysis of Resource Pooling for Network Function Virtualization

Veronica Karina Quintuna Rodriguez and Fabrice Guillemin
Orange Labs
2 Avenue Pierre Marzin
22300 Lannion, France
{veronicakarina.quintunarodriguez,fabrice.guillemin}@orange.com

***Abstract*—In the framework of network function virtualization, we consider in this paper the execution of Virtualized Network Functions (VNFs) in data centers whose computing capacities are limited. We assume that each VNF is composed of sub-functions to be executed on general purpose hardware, each sub-function requiring a random amount of processing time. Because of limited processing capacity, we investigate the relevance of resource pooling where available cores in a data center are shared by active VNFs. We study by simulation various algorithms for scheduling sub-functions composing active VNFs (namely Greedy, Round Robin and Dedicated Core algorithms). We additionally introduce an execution deadline criterion, which means that VNFs can renege if their sojourn time in the system exceeds by a certain factor their service time. This feature is especially relevant when considering the processing of real-time VNF. Simulations show that sub-functions chaining is critical with regard to performance. When sub-functions have to be executed in series, the simple Dedicated Core algorithm is the most efficient. When sub-functions can be executed in parallel, Greedy or Round Robin algorithms offer similar performance and outperform the Dedicated Core algorithm. Enabling as much as possible parallelism and avoiding chaining when designing a VNF are fundamental principles to gain from the available computing resources.**

**Keywords:** Scheduling, virtualization, VNF, cloud computing, fog computing.

## I. Introduction

The emergence of virtualization technology plays a crucial role in the evolution of telecommunications network architectures, notably by enabling the virtualization of network functions. While such functions were so far implemented on proprietary and closed platforms offered by equipment providers, virtualization allows network functions to be implemented on Commercial off-the-shelf (COTS) servers. This is clearly a great revolution in the design of networks and IT infrastructures, which can eventually be completely merged.

For today's network operators, virtualization promises more flexibility and scalability as well as resource efficiency in the deployment of network services [1]. In addition, the use of virtualization techniques allows the network operator to deal with traffic growth without constantly investing in costly infrastructure, since Virtualized Network Functions (VNFs) can be hosted in traditional data centers and customized according to specific needs.

In this context, a VNFs is a piece of software which can even be available as an open source program and executed on general purpose processors (e.g., ARM processors, x86 servers) in multi-platform environments (i.e., operating systems); this guarantees portability and compatibility. Furthermore, virtualization techniques and data center processing enable on-demand provisioning, elasticity and resource pooling. In terms of flexibility and cost efficiency, network functions should not be simply virtualized but also redesigned to gain from the available computing resources.

An end-to-end network service (e.g., video streaming, mobile voice) can be represented as a forwarding graph of network functions, which is commonly referred to as a service chaining [2]. Today's hardware-based approach makes services implementation extremely complex and time-consuming [3]. In a virtualized environment, a VNF Forwarding Graph (VNF FG) runs on the top of the virtualization layer and can be managed more efficiently [2]. For instance, to update a network service, it is enough to add a new VNF on a virtual machine; in the same way, to scale the network, it is only necessary to instantiate the underlying VNF on the computing facility.

Furthermore, each VNF consists of a number of software components, referred to in the following as sub-functions. The flexibility brought by virtualization should be exploited to realize modular and parallel sub-functions. We assume that cores of a computing facility are assigned for executing sub-functions according to a specific scheduling algorithm. This algorithm plays a key role in the system performance, especially when there are real-time constraints or deadlines for the execution of a particular VNF.

In this paper, we shall consider three scheduling algorithms, referred to as Round Robin, Greedy and Dedicated Core. The goal of this contribution is to identify which algorithm is the most efficient in terms of execution time of a VNF and reneging rate (when a deadline for the execution of a VNF applies). In like manner, we aim to determine the impact of VNF design in the global performance of a network function.

The organization of this paper is as follows: Potential use cases are presented in Section II. In Section III, we introduce the model and performance measures to characterize the efficiency of the system. Scheduling algorithms are described

in Section IV. Simulation results of scheduling algorithms with or without reneging are discussed in Section V. Finally, concluding remarks are presented in Section VI.

## II. Problem Statement and Use Cases

Network Function Virtualization (NFV) approach enables the deployment of a large number of VNFs, which can be executed on cloud-computing as well as fog computing environments. While large centralized data centers are commonly operated by the cloud framework, fog computing enables data center dissemination on the edge of the network infrastructure. The advantage of small data centers disseminated at the network edge is that they are close to end-users and ensure lower latency and better customer experience. The key difference between fog and cloud computing is in the limitation of computing capacity [4].

This work pays special attention to the execution of VNFs in data centers with limited capacity in terms of computing. We more precisely investigate the feasibility of resource pooling. We consider a computing facility composed of a number of cores, which can be dynamically allocated for the execution of sub-functions of an active VNF.

Functions of mobile core network (e.g., those of the Evolved Packet Core (EPC) of 4G mobile or future 5G networks) are good candidates for virtualization in the form of virtualized EPC (vEPC), Cloud-EPC (C-EPC) or EPC as a Service (EPCaaS) [5], [6]. As shown in Fig. 1 several functions can be handled by software, such as, session setup, user authentication and access authorization, currently handled by the Mobility Management Entity (MME). This is also the case of packet filters, policy control and charging, supported until now by P-Gateway as well as mobility functions (e.g., in case of handover between eNodeBs or between LTE and other 3GPP access) held by S-Gateway.

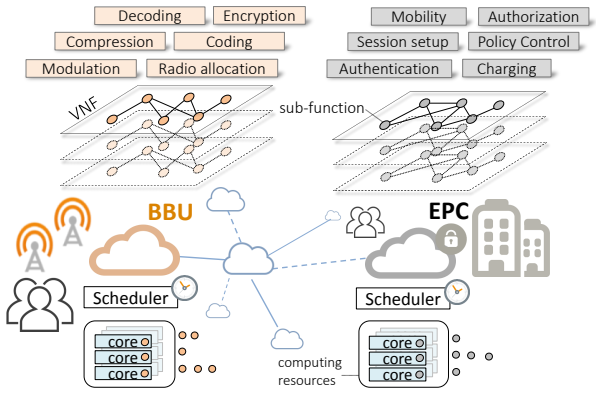


Fig. 1. Network Function Virtualization.

Each VNF can be executed on Virtual Machines (VM) or containers. In this way, operators can instantiate each EPC component in order to cope with the incoming traffic demands. On the other hand, it is also possible to optimize resources during non-peak hours.

Moreover, a virtual EPC could be instantiated for a specific need, e.g., a company which is willing to operate its own private mobile network for its employees. The same applies for other traditional network functions, e.g., IP Multimedia Subsystem (IMS) for private Voice over IP (VoIP) networks.

In spite of the promises of virtualization, several challenges should be dealt with in future implementations, particularly because some network functions have strict requirements in terms of real-time processing. This is notably the case of Radio Access Network virtualization for short vRAN. For instance, in 4G mobile networks, the Time Division Duplex (TDD) process requires a response ACK/NACK (i.e., ACK to proceed to the transmission of a new frame, or NACK to attempt a re-transmission) to the User Equipment (UE) or eNodeB within 3ms after a frame is arrived, where 2ms are available for reception process and 1ms for transmission process [7]. This operation named Hybrid Automatic Repeat-Request (HARQ) is performed at the MAC level, after demodulation and decoding are done. If successful decoding is not possible the packet re-transmission is scheduled. Nevertheless, this fact degrades the customer experience, because it reduces the achievable UE peak rate [8].

Considering the realization of vRAN, traditional eNodeBs with integrated radio and base-band processing are replaced with shared processing and distributed radio elements, i.e., Remote Radio Head (RRH). So, virtual Base Band Unit (BBU) sub-functions are located at a centralized site, named below vBBU, where a pool of computing resources could be dynamically allocated based on traffic conditions. In addition, BBU virtualization can dramatically improve the global network performance; since several radio elements are handled by a single BBU pool, inter-cell coordination facilitates cooperative multi-point processing and massive MIMO implementation while avoiding interference [9].

In this way, the entire BBU functionality [10], [11] including RAN L1, L2, and L3 protocol layers could be represented as a suite of virtual sub-functions available for treating Physical Resource Blocks (PRB), as illustrated in Fig. 1. Modulation and encoding process as well as data convergence process (e.g., packet compression and encryption) and radio bands allocation might be considered as virtual sub-functions of the base-band processing module. Thus, the processing load mainly depends of the Modulation and Coding Scheme (MCS), the execution time of these sub-functions is determined by the channel conditions between the eNodeB and the current UE. Depending on which value is reported by the UE, network transmits data with different MCS. If network gets high Channel Quality Indicator (CQI) from UE, (i.e., good radio quality) it transmits the data with high-order modulation and a simple channel coding scheme. Likewise, for a low CQI, the network constructs the transmission block with a robust MCS. All sub-functions except modulation/demodulation and encoding/decoding might have small run time enabling the possibility of sharing computing resources [8].

Decoding function (e.g., LDPC, turbo codes) is the most complex in terms of processing load, since its number of iterations depends on the signal quality. Therefore, decoders are promising candidates for parallelization to meet real-time

constraints; for instance, multiple code words may be decoded in parallel or even the decoder itself might be decomposed into multiple threads that run in parallel [8].

In view of the two above use cases (i.e., vBBU, vEPC), we see that virtualization requires an appropriate decomposition of virtual network functions in order to guarantee an efficient resource utilization in terms of computing by adequate scheduling algorithms. This fact enables resource pooling and statistical multiplexing in the execution of sub-functions on multi-core platforms, performing the same tasks with less hardware or capacity. It is set out in more detail in the next section.

## III. Model Description

### A. Model setting

At present resource allocation in virtual platforms have near static behavior; virtual machines or containers are reserved for a specific VNF (e.g., vEPC, vBBU) even though a VNF is sporadically invoked. As a consequence, efficiency in resource utilization is not achieved, since computing resources are frozen but no used. It would thus be better to perform statistical multiplexing on computing resources. More precisely, in the following, we assume that a set of cores is available to execute VNFs with dynamic resource orchestration. Cores are allocated for the execution of a VNF when that function is invoked.

As discussed in Section II, in most cases, the runtime of a virtual network sub-function is deterministic. Nevertheless, other aspects discussed in [12] are intrinsic to the execution of a sub-function such as memory access time, caching policies, disk access time, inter-processor communication, system buses and I/O buses behavior, among others. In fact, the multiprocessing architecture plays an important role in the whole network virtualization performance. In the following, we include these issues in the holding time of cores, which becomes a random variable. This is a classical assumption in network performance modeling when various factors influence the execution of a job.

Taking into account a pool of cores which are statistically shared by several active VNFs, a significant gain in resource savings is expected. The counterpart is that the execution of a VNF might be delayed until some core is available in comparison with the case when computing resources are dedicated to a single VNF. In this paper, we assume that a VNF is composed of sub-functions, each of them being executed on the multi-core platform. The goal of this work is to investigate how the underlying sub-functions should be scheduled or even conceived to improve the VNF performance.

In this context, the functional disaggregation in the virtualization process should take into account the correspondence between sub-functions, because it determines the behavior in the execution process. Then, a particular VNF could be viewed as a process flow with sub-functions either running in sequence or else being executed in simultaneous threads. The present work analyzes the behavior of scheduling algorithms for both configurations, i.e., when VNFs are executed as a chain of sub-functions in contrast with the case when sub-functions are allowed to be executed in parallel.

### B. Queuing system formulation

Let us consider a computing facility (i.e., a data center) equipped with a pool of $c$ cores capable of executing elementary sub-functions. These sub-functions are part of a VNF, also referred to macro-function. Such a VNF is composed of $k$ sub-functions; when a VNF is invoked the entire batch of $k$ sub-functions has to be handled by the computing facility. The model proposed below takes $k$ as a constant, however, it could be easily extended to the case when $k$ is a random variable (e.g., with a geometric distribution).

VNF requests occur at the computing facility according to a Poisson process with rate $\lambda$. The execution time of a sub-function is random with an exponential distribution, with mean $1/\mu_j$ for the $j$th sub-function of a VNF, where $j = 1, \ldots, k$. We assume that execution times of various sub-functions are independent, hence, the execution time of a macro-function is originally the sum of $k$ exponential random variables. The mean execution time is

$$\mathbb{E}[s] = \sum_{j=1}^{K} \frac{1}{\mu_j}.$$

When all $\mu_j$ are equal to some $\mu > 0$, the execution time is simply an Erlang random variable with mean $k/\mu$ and variance $k/\mu^2$, denoted, for short, by $\mathrm{Er}(k, \mu)$.

In general, this model involves a pool of cores which execute only one sub-function at a time. The holding time of a core by the $j$th sub-function of a VNF is exponential, with mean $1/\mu_j$. Hence, when a request occurs while all cores are busy, the function is queued, see Fig. 2 for an illustration. The total amount of time $r$ to treat a VNF by the computing facility is composed by the queuing delay $q$ and the execution time of sub-functions $s$, so that, the system response time for the execution of a VNF is given by $r = s + q$. The queuing delay is the amount of time that the VNF spends in the system while none of its sub-functions is executed.

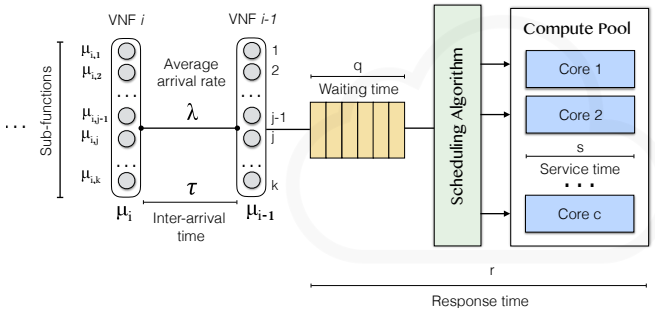


Fig. 2. Elements of the queuing system.

We assume that the waiting room of the computing facility is infinite. It is then clear that the system is stable if and only if the load $\rho$ defined by

$$\rho = \frac{\lambda \mathbb{E}[s]}{c}$$

is less than 1. In the following, we assume that this stability condition is always satisfied. That condition is not necessary in the case of reneging, since this phenomenon makes the system stable at the price of rejecting requests.

### C. Performance measures

In view of the above queuing system formulation, we characterize its performance by the distribution of the sojourn time $r$ of VNFs in the system. If $r$ takes small values, then macro-functions (VNFs) are rapidly executed and consequently, the overall efficiency is achieved.

On the other hand, if the sojourn time is too large, then some VNFs are slowly executed and in some cases (e.g., vRAN) completing the execution of a VNF becomes useless. As presented in Section II, a time budget of 3ms for BBU processing is available to allow continuous transmission per UE. This fact is equivalent to introduce the concept of reneging. This budget represents the time between the end of the up-link transmission and the start of the down-link subframe carrying the corresponding ACK/NACK [13]. More precisely, a customer reneges if its sojourn time in the system exceeds too much its service time. Reneging in multiple-servers queuing systems is discussed in [14]. The adopted reneging criterion is given below:

*Definition 1 (Reneging macro-function):* A VNF execution request reneges if its sojourn time $r > (1 + \theta)s$ for some $\theta > 0$. This is equivalent to the condition $q > \theta s$, where $q$ is queuing delay of the macro-function.

In case of reneging, the VNF leaves the system and all resources used prior to its departure are wasted. Hence, the performance of the system is characterized beyond the sojourn time $r$ by the reneging rate $\eta(\theta)$ as a function of the parameter $\theta$.

Whether with the distribution of sojourn time or reneging, the system performance depends on the queuing delay of VNF requests, i.e., the time that a VNF spends in the system without receiving service. This queuing delay depends on how sub-functions are sorted for their execution. This is precisely the role of the scheduling algorithms described in the following section.

## IV. Scheduling algorithms

The scheduling algorithm determines which VNF and, more precisely, which sub-function gets access to a particular core of the computing facility. In other words, the scheduling algorithm selects which sub-function is executed. The scheduling strategy has a direct impact on the system response time as well as on the resources utilization efficiency. Thus, the scheduling analysis aims at identifying conditions and constraints to improve the system performance.

The scheduler selects the next sub-function to be processed among VNFs that are ready to be executed, and allocates to it a processing unit (core). Let us consider three scheduling algorithms: Dedicated Core (DC); Round Robin (RR) and Greedy (G) presented in the following subsections.

*1) Allocating the entire macro-function to a Dedicated Core:* This method processes all sub-functions forming a particular VNF upon the same core. The scheduler selects the VNF keeping the arrival order. In this case, the computing facility can be described by an $M/G/c$ queue system, where the service time is the sum of independent exponential random variables. If all $\mu_j$ are the same, then an $M/\mathrm{Er}(k,\mu)/c$ queue system is obtained.

*2) Allocating sub-functions by Round Robin criterion:* This algorithm handles the VNFs under the RR criterion. Incoming VNFs integrate the circular cycle upon arrival, while the scheduler selects them in first-in-first-out order. Hence, sub-functions are assigned to the pool of cores in a circular order. As detailed in [15], Round Robin generally employs time-sharing, assigning to each task a time slot, and interrupting the task if it is not completed. In this paper, the proposed algorithm does not force the sub-function going out of the core if it is not finished, since we assume that sub-functions are not divisible.

*3) Greedy allocation of sub-functions:* Unlike the precedent algorithms, the arrival order of VNFs is prioritized at the moment of allocating sub-functions, in the sense that, if a VNF starts its service, then the system tries to finish it as soon as possible. Thus, when the first sub-function of a VNF starts its execution, the next core which becomes available is used to serve the second sub-function and so on.

As a consequence, those VNFs which have started their service have priority over those which have not begun their execution. Incoming VNFs are obliged to wait to be processed until precedent VNFs have completely finished their execution. In other words, the system does not try to share the computing resources among the VNFs which are in the system as in the case of RR algorithm. Greedy algorithm aims at completing the service of VNFs at the earliest possible way without fairness.

## V. Performance Analysis

### A. Simulation setting

The fundamental issue is to analyze the performance of the scheduling algorithms introduced in the previous section. We are interested in the sojourn time distribution of VNFs as well as in the reneging rate in case of deadline for the execution of a VNF. Keeping in mind that a VNF is formed by sub-functions, which can be executed in parallel or in series, the behavior of scheduling algorithms is analyzed for both cases.

Different scenarios have been considered in relation with the computing pool size and the number of sub-functions per VNF, taking values where $c < k$, $c > k$ and $c \gg k$. Furthermore, we have considered different load conditions, namely $\rho = 0.6$ (moderate load) and $\rho = 0.9$ (heavy load). Finally, we analyze the behavior when all sub-functions have the same mean execution time. This means that for all $j = 1, \ldots, k$, $\mu_j = \mu$ for some constant $\mu > 0$. The parameter $1/\lambda$ equal to the mean inter-arrival time of VNFs is taken as time unit. The parameter $\mu$ is adjusted so that the load $\rho = k\lambda/(\mu c)$ is equal to a prescribed value.

The performance results analyzed below correspond to those scenarios with heavy load and a number of cores greater than the number of VNF's sub-functions. The same configuration was applied when sub-functions belonging to a particular VNF have different mean execution times. We have also evaluated the performance when different classes of VNFs share the same multi-core platform. This latter scenario offers the possibility to support new services with the same infrastructure, having for instance the co-execution of RAN functionality with other network functions. The results obtained for these scenarios are qualitatively the same as those presented below and are not reported in this paper.

### *B. Scheduling performance without reneging*

In the framework of VNFs where their components represent a forwarding graph of sub-functions, Greedy algorithm is not applicable because the chaining constraint can not be respected. For instance, BBU is a chained process whose functional blocks roughly are modulation/demodulation, encoding/decoding, radio scheduling, and RLC/MAC PDU generation/decomposition. Hence, we analyze the performance in terms of sojourn time of VNFs when these are scheduled following the RR criterion as well as the DC algorithm. Fig. 3 illustrates the behavior of both algorithms considering a highly loaded system with parameter $\rho = 0.9$, which means that the arrival rate of VNFs is important for the computing capacity.

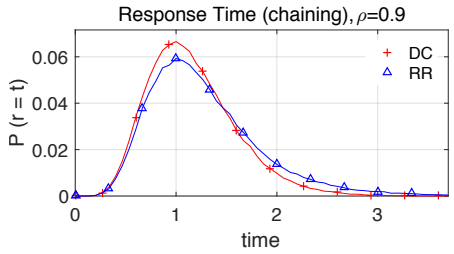


Fig. 3. Scheduling performance considering chained sub-functions

From Fig. 3, it turns out that the simple DC algorithm offers slightly better performance than the RR algorithm, especially for the tail the of the sojourn time distribution. This will be confirmed by considering the case when VNFs can renege. The additional delay introduced by a higher sojourn time represents in most cases the degradation of the customer experience. For instance, when executing a vBBU, all information sent from the physical layer to the MAC layer and vice-verse has to deal with this extra time. In this case, a critical effect of a high sojourn time is the caducity of channel measurements sent from the UEs to the eNodeB, which are used for radio scheduling and other fundamental issues in the characterization of radio signals. As a consequence, the mobile network losses both energy and spectral efficiency.

Let us consider now, VNFs whose sub-functions are allowed to be executed in parallel. This means that the execution of a particular sub-function is independent of the results of the previous one. In this context, Greedy (G) algorithm can be applied. Fig. 4 presents the behavior of three algorithms where the performance of both G and RR is notably better than the behavior obtained with the DC algorithm. Even more, G algorithm shows a slightly better performance than that of RR.

It is evident that the chaining constraint considered in the first simulation balks the advantages of RR algorithm in its attempt of fairness. In this way the simplicity embedded in DC criterion is the most appropriate for a computing pool environment, all the more as the complexity of RR does not improve the sojourn time of VNFs.

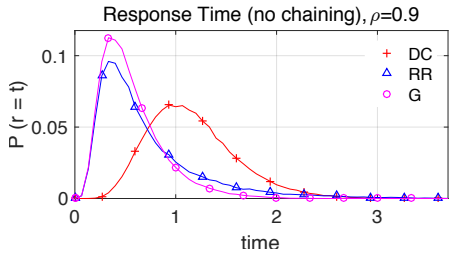


Fig. 4. Scheduling performance considering no chained sub-functions

### *C. Scheduling performance considering a deadline in the execution of VNFs*

In this subsection, we analyze the scheduling performance considering reneging with parameter $\theta = 1$. As explained in Section III, this factor represents the deadline present in some network functions which require real-time execution as in the case of base band processing of mobile networks. Considering the scenario where $c > k$ and applying the chaining constraint, Fig. 5 illustrates the behavior of RR and DC algorithms. Again their performances in terms sojourn time of a VNF are similar with an advantage for DC when considering the tail of sojourn time distributions.

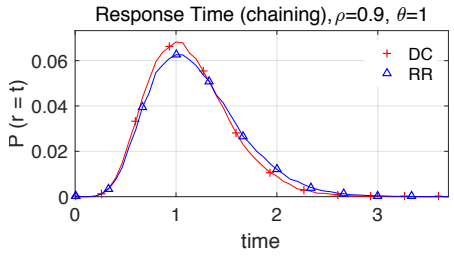


Fig. 5. Scheduling performance considering chained sub-functions and reneging

Table I shows reneging rate which represents the number of VNFs that have not finished their execution.

The utilization rate and waste rate show respectively the occupation of the computing platform by VNFs which have been completely processed and the occupation of cores by sub-functions belonging to VNFs which have reneged. DC offers a slightly better performance, in line with the observation made for the tail of the sojourn time distributions when there is no reneging.

TABLE I
SCHEDULING PERFORMANCE WITH CHAINED SUB-FUNCTIONS

| | Reneged Rate | Utilization Rate | Waste Rate |
|---|---|---|---|
| DC | 1.6270 | 99.0821 | 0.6816 |
| RR | 4.5060 | 97.0478 | 2.4837 |

RR algorithm yields a higher reneging rate, and consequently worse utilization factor than DC algorithm. Hence, more computing resources are wasted.

Analyzing the case when sub-functions can be executed in parallel, the behavior of scheduling algorithms turns out again more favorable for G and RR criteria. It is shown in Fig. 6 where reneging rate was established with $\theta = 1$, it means that a VNF interrupts its service and leaves the system if its sojourn time is greater than the double of the required execution time. The same kind of behavior has been observed for smaller and greater values of $\theta$.

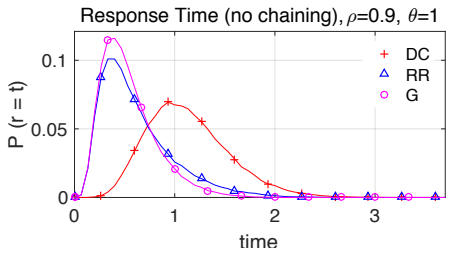


Fig. 6. Scheduling performance considering no chained sub-functions and reneging

As much as in terms of execution delay as in terms of reneging rate presented in Table II, G has the best performance.

TABLE II
SCHEDULING PERFORMANCE WITH NO CHAINED SUB-FUNCTIONS

| | Reneged Rate | Utilization Rate | Waste Rate |
|---|---|---|---|
| DC | 1.6220 | 99.0478 | 0.6956 |
| RR | 2.6430 | 98.3832 | 1.2734 |
| G | 0.4560 | 99.7832 | 0.0625 |

It is evident that the behavior of DC is relegated by RR and G performance. Nevertheless, the utilization rate of DC remains interesting when compared with that resulting of RR execution. Results show that the worst algorithm in terms of resources saving is RR although it keeps a better sojourn time. Greedy becomes the most efficient and notably the most suitable when the execution of sub-functions is not limited by the chaining constraint.

## VI. CONCLUSION

In this work, we have studied a system executing VNFs on a computing platform composed by a pool of cores; each VNF is composed of several sub-functions. We have analyzed by simulation the performance of three algorithms for scheduling the execution of sub-functions of active VNFs (namely, Round Robin, Dedicated Core and Greedy). It turns out that when sub-functions can be executed in parallel, the Greedy algorithm ensures the best performance in terms of execution delay. We have also considered the case when VNFs may renege because the sojourn time in the system exceeds some threshold related to the required amount of service. Still in this case, the Greedy algorithm offers the best performance.

In the case of chained sub-functions Greedy algorithm can not be applied, and the performances observed with Dedicated Core and Round Robin are similar, so the complexity added by this latter is not justified.

This phenomenon has to be taken into account when designing VNFs executed on a pool of cores. In particular when decomposing a network service into components or micro-services, the best choice is to decompose a function into sub-functions which can be executed in parallel and independently of each other.

## REFERENCES


[1] Slavisa Aleksic and Igor Miladinovic. Network virtualization: Paving the way to carrier clouds. In *Telecommunications Network Strategy and Planning Symposium (Networks)*, pages 1–6, 2014.

[2] Network Functions Virtualisation ETSI. Architectural framework. Technical report, Technical Report ETSI GS NFV 002 V1.1.1, 2013.

[3] ONF Solution Brief. Openflow-enabled sdn and network functions virtualization. *Open Netw. Found*, 2014.

[4] Ivan Stojmenovic and Sheng Wen. The fog computing paradigm: Scenarios and security issues. In *Computer Science and Information Systems (FedCSIS), 2014 Federated Conference on*, pages 1–8. IEEE, 2014.

[5] Tarik Taleb, Marius Corici, Carlos Parada, Almerima Jamakovic, Simone Ruffino, Georgios Karagiannis, and Thomas Magedanz. Ease: Epc as a service to ease mobile core network deployment over cloud. *Network, IEEE*, 29(2):78–88, 2015.

[6] Kostas Pentikousis, Yan Wang, and Weihua Hu. Mobileflow: Toward software-defined mobile networks. *Communications Magazine, IEEE*, 51(7):44–53, 2013.

[7] I Chih-Lin, Jinri Huang, Ran Duan, Chunfeng Cui, Jesse Xiaogen Jiang, and Lei Li. Recent progress on c-ran centralization and cloudification. *Access, IEEE*, 2:1030–1039, 2014.

[8] Dirk Wubben, Peter Rost, Jens Steven Bartelt, Massinissa Lalam, Valentin Savin, Matteo Gorgoglione, Armin Dekorsy, and Gerhard Fettweis. Benefits and impact of cloud computing on 5g signal processing: Flexible centralization through cloud-ran. *Signal Processing Magazine, IEEE*, 31(6):35–44, 2014.

[9] Alexander William Dawson, Mahesh K Marina, and Francisco J Garcia. On the benefits of ran virtualisation in c-ran based mobile networks. In *Software Defined Networks (EWSDN), 2014 Third European Workshop on*, pages 103–108. IEEE, 2014.

[10] Navid Nikaein. Processing radio access network functions in the cloud: Critical issues and modeling. In *Proceedings of the 6th International Workshop on Mobile Cloud Computing and Services*, pages 36–43. ACM, 2015.

[11] Rui Wang, Honglin Hu, and Xiumei Yang. Potentials and challenges of c-ran supporting multi-rats toward 5g mobile networks. *Access, IEEE*, 2:1187–1195, 2014.

[12] Michael Haugh. Examining factors of the nfv-i impacting performance and portability. 2015.

[13] Uwe Dötsch, Mark Doll, Hans-Peter Mayer, Frank Schaich, Jonathan Segel, and Philippe Sehier. Quantitative analysis of split base station processing and determination of advantageous architectures for lte. *Bell Labs Technical Journal*, 18(1):105–128, 2013.

[14] Liqiang Liu and Vidyadhar G Kulkarni. Balking and reneging in m/g/s systems exact analysis and approximations. *Probability in the Engineering and Informational Sciences*, 22(03):355–371, 2008.

[15] Ishwari Singh Rajput and Deepa Gupta. A priority based round robin cpu scheduling algorithm for real time systems. *International Journal of Innovations in Engineering and Technology*, 1(3):1–11, 2012.