Understanding the performance of flexible functional split in 5G vRAN Controllers: A Markov Chain-based model

Luis Diez, Cristina Hervella and Ramón Agüero *Senior Member, IEEE*
Communications Engineering Department
University of Cantabria
{ldiez, ramon}@tlmat.unican.es,
cristina-aurora.hervella@alumnos.unican.es

We study Flexible Functional Split functionality of 5G vRAN controllers in 5G networks. We propose an innovative model, based on a Markov Chain, which can be used to characterize their performance. We consider both infinite and finite-buffer controllers. In the former, frames would not be lost (provided the system works in a stable regime), and we thus focus on the time frames stay at the controller. For the finite-buffer controller, there might be losses, and we analyze the trade-off between time at the controller (which might hinder the stringent delay requirements of 5G services), and loss probability. Matrix-geometric techniques are used to resolve the corresponding Quasi-Birth-Death process. The validity of the proposed model is assessed by means of an extensive experiment campaign carried out over an ad-hoc event-driven simulator, which is also used to broaden the analysis, considering different service rate distributions, as well as the variability of the studied performance indicators. The results show that the proposed model can be effectively exploited to tackle the dimensioning of these systems, as it sheds light on how their configuration impacts the expected delay and loss rate.

***Index Terms*—Flexible Functional Split, vRAN, Controller, Markov Chain, Quasi-Birth-Death Process**

I. Introduction

The pervasive presence of 5G communications requires networks to be adaptive and highly customizable. Indeed, 5G requirements are usually categorized to serve three main service types, with rather distinct features. On the one hand, enhanced Mobile Broadband (eMBB) extend traditional wireless communication services, requiring much more capacity, as well as ubiquitous connectivity. In addition, Quality of Service (QoS) of novel service types, such as massive Machine Type Communications (mMTC) and Ultra Reliable Low Latency Communications (URLLC), is based on different metrics. This service diversity imposes future networks to be adaptable and re-configurable, to adequately satisfy the required communication performance.

Over the years, Self-organizing Networks (SON) solutions have been proposed to make wireless communication networks more responsive and adaptable to both failures and varying requirements. In this regard, virtual RAN (vRAN) [1] is postulated as the main transformation of 5G networks to enable SON implementation [2]. Under the vRAN paradigm, which exploits Software Defined Networking (SDN) and Network Function Virtualization (NFV) techniques, traditional functionalities of the Radio Access Network (RAN) are virtualized, and the logical Base Station (BS) is divided in different functional entities. Then, the centralization of the virtualized entities allows a tight cooperation among the different access elements, so improving the network performance.

In order to develop the vRAN concept, initially fully centralized solutions, Cloud RAN (C-RAN), were proposed [3], [4]. In C-RAN the Base-Band Unit (BBU) of the base station is fully virtualized and centralized, while a Remote Radio Head (RRH) stays at the particular site position, together with the antenna and basic physical layer functions. Although this approach would yield a tight coordination among the access elements, it also demands high capacity in fronthaul links [5], which connect RRHs and virtualized BBUs, only affordable with a vast deployment of costly fiber links.

To overcome this limitation, industry, academia and standardization bodies have been working together to define different virtualization levels, so called functional splits [2], [6], with various fronthaul communication requirements. The reader may refer to work of Checko *et al.* [7] for a thorough discussion of the impact of different functional splits, in terms of energy, QoS and cost. This way, according to the configured functional split, part of the RAN functions are virtualized in a Central Unit (CU), typically located close to the transport network edge. The remaining functions are kept close to the access network edge, leading to the so-called Distributed Unit (DU) [8], while the RRH is now also referred to as Radio Unit (RU). More recently, some initiatives have suggested a flexible split selection, so that the virtualized functions in the DU and CU can be dynamically shifted.

One of the main challenges of this new architecture is the management of the fronthaul segment, which goes from being a set of dedicated links to a more complex packet based network, that needs to be appropriately managed. In fact, relevant standardization initiatives have started to define the Next Generation Fronthaul Interface (NGFI) [8], [9]. The NGFI is divided into NGFI-I and NGFI-II, or midhaul, which communicates RUs with DUs, and DUs with CUs, respectively. Altogether, it becomes necessary to develop solutions that jointly consider routing through the fronthaul network, along with the potential dynamic selection of the functional

split that satisfies the required QoS. In turn, the split selection would be constrained by the computational resources allocated to the DU. In this sense, an adequate model of controllers, able to capture all these parameters and their impact over the corresponding performance, would be fundamental to tackle the dimensioning of future access networks. It is worth noting that the service chain in vRANs introduces constraints in the way functions are embedded. In particular, the DU and CU host lower and upper layers of the protocol stack, respectively. Thus, opposed to other NFV areas [10], the vRAN service-chain has only a limited set of possible implementations (i.e. a given frame would not go forth and back between elements to traverse the protocol stack).

In this paper, we present a controller model for vRAN architectures, which considers both the split selection and computational resources. We do not aim at proposing a particular split selection strategy to optimize the controller behavior, but at building a model able to assess its performance under a given split strategy. More precisely, assuming a particular flexible split selection policy and allocated processing resources, the model provides expected performance of the system, in terms of delay and loss probabilities. Exploiting Markov Chain theory, we model the flexible functional split controller as a Quasi-Birth-Death (QBD) process. We consider two types of controllers: (1) infinite-buffer, where the buffer at the CU is considered to have enough capacity, so frames would not be lost; and (2) a finite-buffer controller, where frames could be discarded, when the buffer gets full. To our best knowledge, this is the first attempt to develop a theoretical model of CU controllers to appropriately analyze the performance of 5G flexible functional split architectures. We note that, although in this work we apply the model to the CU, it could be also used to evaluate the performance of the DU.

The contributions of this work can be summarized as follows:

- We model the controller of the flexible functional split architecture, both assuming infinite and limited buffer capacity at the CU.
- The model is based on a bi-dimensional Markov Chain, which boils down to a QBD, and that is solved with matrix-geometric techniques.
- We validate the model, by comparing the theoretical results with those obtained by an ad-hoc event-driven simulator, in terms of delay, and loss probability.
- The simulator is also exploited to broaden the analysis, considering different service time distributions, as well as studying results variability.
- The implementation carried out to obtain the results presented in this paper, Matlab scripts and event-driven simulator (C++), have been made available to the community in a public repository[1].

The rest of the paper is structured as follows. In Section II we discuss the related work, pointing out the differences between our research and existing literature. Then, we depict the controller model as a Quasi-Birth-Death process in Section III, and in Section IV we theoretically find its main performance indicators. In Section V we validate the model, comparing its results with those obtained with an event-driven simulator. The paper concludes in Section VI, where we summarize its main outcomes, and we provide an outlook of our future work.

[1]https://github.com/ldiez/5GvRanControllerQBD

II. Related work

In recent years, a number of studies have tackled flexible functional split from different angles. In [11], the roadmap to implement flexible functional splits solutions is described, while the main characteristics of this kind of architectures are defined in [12] and [13]. Similarly, Arnold *et. al.* describe a two-level flexible functional split architecture in [14].

Other works have focused on implementation details, thoroughly analyzing their impact. In this sense, in [15] and [16] Chang *et al.* studied the interplay of split selection with packetization and scheduling, respectively. Furthermore, compressing and coding solutions are proposed in [17], to enable the implementation of flexible split selection. In a more generic way, an implementation of a flexible functional split framework, along with its performance analysis, can be found in [18], and a real-time demonstrator is described in [19].

If we focus on the split selection policies, a number of works have recently analyzed the requirements and proposed solutions. In this regard, Martínez Alba and Kellerer [20] analyzed the performance and requirements of convergence times of split selection algorithms. One of the most common constraints of the split selection policy is related to delay. Indeed, some works pay special attention to such performance indicator. For instance, reinforcement learning is used in [21] to enforce delay constraints in dynamic split selection. Besides delay, other works jointly combine split selection with other relevant parameters for network management and service provisioning. For instance, a joint split selection and content caching solution is proposed in [22]. Some works have also put emphasis on the impact that this type of architectures might have over energy consumption. In this sense, split selection is optimized considering energy constraints in [23], [24], while the authors of [25] propose an energy-aware split selection algorithm for scenarios with Unmanned Aerial Vehicles (UAVs). In the same line, the selection of functional splits in [26], as well as their duration, considers energy constraints over a network with energy harvesting-enabled radio elements. In contrast, the authors of [27] propose a novel solution to optimize split selection taking into account energy and handover strategies.

In addition, some works have studied the implementation of split selection over optical fiber networks [28]–[32]. For instance, the authors of [31] depict an architecture, called F-RAN, able to dynamically choose the best split selection, based on radio performance and optic transport capacity. The architecture employs SDN to manage the underlying optical network, according to the selected split. Similarly, the authors of [32] propose a two-step solution to adapt the transmission of the underlying optical network, when degradation in the light-path occurs, along with split selection reconfiguration. Other works, such as [33], have also analyzed the interplay between dynamic split selection and network slicing.

Worthy of attention are those works that combine functional split selection policy and routing in the fronthaul network. The authors of [34] propose a framework that integrates heuristic solutions to optimize energy efficiency in converged fronthaul/backhaul networks. Although the work does not directly tackle flexible functional split, it is mentioned that reallocation of virtual functions can be done along with the optimization. As an example, an heuristic solution to dynamically allocate new traffic flows is described in [35]. Similarly, Li *et al.* provide in [36] an architectural view of resource management and energy efficiency for converged fronthaul/backhaul networks. In particular, for the resource management, the authors jointly consider path computation and placement of network functions. Besides, an heuristic algorithm, based on backtracking branch and bound, is proposed in [37] to solve the joint split selection and routing problem. This scenario is further extended in [38], where placement of Mobile Edge Computing (MEC) elements is included, and an algorithm based on Benders decomposition is proposed to solve the resulting problem in [39]. While the previous work considers that the location of the CUs is known, placement of CUs, along with path selection, is included in the optimization problem. In this case, the split configuration is not a direct selection, but a consequence of solving the network embedding problem.

Most of the aforementioned works propose split selection policies, either standalone or in combination with other parameters and techniques, but they are rather different from our work in nature. Our main goal is to define a model that provides expected controller behavior when a particular policy is applied. Based on this model, we could obtain a more accurate expected performance, which could be exploited to assess the mentioned proposals, by using the appropriate statistical behavior of the corresponding split policies.

Another group of works take into account computation capabilities in the controller when performing the split selection, along with other network management techniques. Koutsopoulos [40] proposes a set of heuristics for flexible functional split solutions that minimize the delay, considering the controller processing capacity. Furthermore, an scheduling solution for vRAN is proposed in [41], which ensures that computation resources in the controller do not get exhausted. To this end, the authors propose an adaptive Modulation and Coding Scheme (MCS) solution when computation resources cannot satisfy the demand.

A machine-learning based solution is proposed in [42] to manage computation resources in the vRAN controller. In addition, the authors carefully analyze the computational load required by the different network functions, using a Software Defined Radio (SDR) implementation. Following this line, Rost *et al.* [43] propose two computation-aware schedulers to maximize the sum-rate, while respecting computation limits. Energy is also considered in [44], where resource allocation in vRAN is jointly tackled with Mobile Cloud Computing (MCC), to perform user tasks. In particular, authors model the BBU to consider the impact of computational capacity on radio resource scheduling. A similar scenario can be found in [45], where task offloading in MEC is tackled, considering flexible functional split. As can be observed, our work clearly differs in its scope. Although these papers consider computational resources, none of them address an appropriate modeling of the vRAN controller.

Furthermore, we have also identified two works that aim to model some aspects of the virtualized network. The authors of [46] provide a quantitative analysis of computation and power savings when using different splits. In particular, the authors seek to estimate processing (power) reduction in the CUs pool when applying different splits. In contrast, we focus on networking and computational metrics related to QoS, such as delay, buffer size, and loss probability. Finally, a queue-based model for packet generation using SDN techniques is proposed in [47]. Although this model might be applied to improve the design of the fronthaul network, by tweaking the corresponding traffic patterns to deal with congestion, the scope of this work is different to ours, which focuses on a model to yield key performance indicators of the vRAN controller.

We can thus conclude that, to our best knowledge, the modeling of the vRAN controller has not been so far tackled. The model we propose herewith is able to provide accurate expected performance, considering key indicators for 5G services, such as delay and loss probabilities. It can be thus exploited to address the dimensioning and planning of this type of network elements, since it can be configured to consider different split selection policies.

III. Controller Model

We consider a vRAN controller, which is equipped with a single CU, having a certain number of processing and memory resources. Downlink communication is assumed, although the model can be straightforwardly applied for uplink traffic. Frames towards the CU arrive at a rate of $\lambda\ ms^{-1}$, following a Poisson process. The CU can be configured in s different splits, each of them having a certain service rate $\mu_k\ ms^{-1}$ for $k \in \{1, ..., s\}$. We assume that service times follow an exponential distribution, and that a First-Come First-Served (FCFS) policy will be used to process frames.

In the following we define two CU models. In the first one, we will assume that the buffer has enough capacity to store frames before they can be eventually served, provided that it operates at a stable regime. In this case, frames will not be lost. On the other hand, we will also assume a more realistic setup, where the buffer has a finite capacity and so frames could be eventually discarded.

In both cases, the controller can use different split configurations. We assume that a split change is performed after a certain time interval, which is modeled as an exponential random variable, with rate $\gamma\ ms^{-1}$. Once this timer expires, the controller goes into a stand-by state, where frames are not served, until it becomes again operative. Frames may continue coming, but they are kept in the buffer (if there is capacity left). We assume that the CU leaves such stand-by state after a time, which is also modeled with an exponential random variable, with mean value $\xi^{-1}\ ms$. Upon this timer expiration, the next configuration split is randomly selected, so that split k^{th} is selected with a probability α_k, and $\sum_{k=1...s} \alpha_k = 1$. It is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

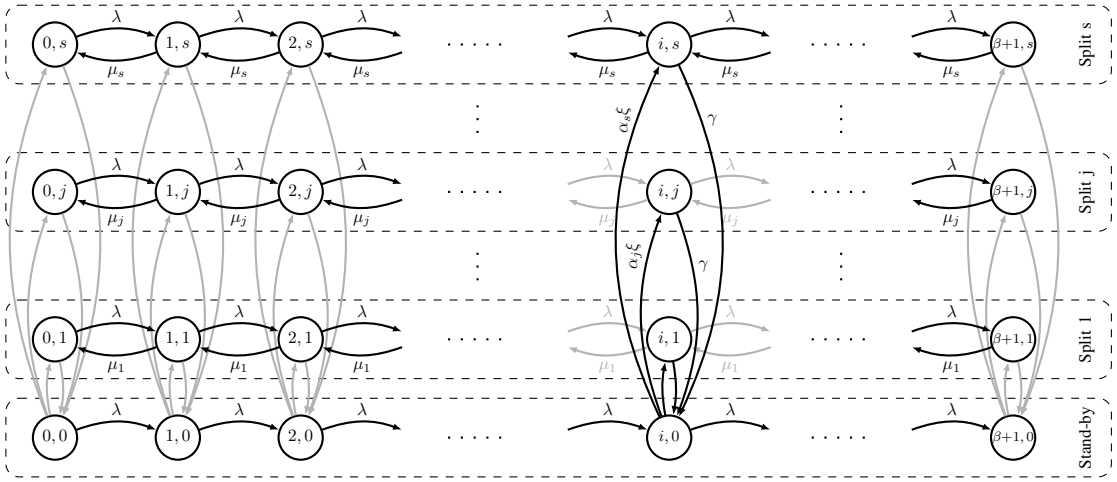


Fig. 1: Markov chain for the controller model, with a buffer of capacity β frames

TABLE I: Model variables

s	Number of available slit configurations
β	Buffer length
λ	Frame arrival rate
μ_k	Service rate of the k^{th} split
α_k	Probability of using the k^{th} split $\left(\sum_{i=1}^{s} \alpha_k = 1\right)$
γ	Split change rate
ξ	Inverse of time at stand-by
$\pi_i(j)$	Probability of state (i,j) (having i frames when CU is using split j)
π_i	Column vector: $[\pi_i(0) \dots \pi_i(j) \dots \pi_i(s)]$
$\mathcal{Q},\ \mathcal{Q}_b$	Infinitesimal matrices of the QBD processes
F	Forward transition matrix
B	Backward transition matrix
$L, L_0, L_{\beta+1}$	State transition matrices within the same level

worth noting that split transition rates and probabilities would be defined by the particular split selection policy used. On the other hand, the time spent in the stand-by operation state would depend on computational resources given to the DU and the particular software implementation of the vRAN solution.

We model the state at which the controller is currently operating as a tuple (i, j), where i is the current number of frames at the controller, while j corresponds to the current split. Therefore, if j equals 0, the CU would be at stand-by. Table I enumerates all the symbols that are used in the proposed model and its solution.

A. Markov chain

As we have considered that all services are modeled by means of exponential random variables, and that frames arrive at the controller following a Poisson process, we can use the 2-dimensional Markov Chain shown in Figure 1 to model the behavior of the CU. As can be seen, each of the rows corresponds to a particular split, where a frame arrival leads to a 1-state rightwards transition, and a frame exiting the system (after it has been completely served) is captured by a 1-state leftwards transition. In the figure we have assumed that the controller has a finite buffer, which can keep up to β frames. In this case, if $i = \beta + 1$ (last column at the right side of the Markov Chain), the controller would not accept any incoming frame, which would be lost, and the only possible transition happens when a frame completes its processing (1-state leftwards). If we assumed that the controller would have enough capacity to maintain incoming frames until they could be eventually processed (this requires system stability), the corresponding Markov Chain would have infinite states, and we would not strictly need such last column $(\beta + 1, k)$ in the corresponding diagram.

Whenever the DU changes its split configuration, it first goes to the stand-by mode. As was mentioned earlier, we model these split changes with a exponentially distributed timer, whose mean value is $\frac{1}{\gamma}\ ms$. This is reflected by the transition from each state (i, k), with $k = 1, \dots s$ to $(i, 0)$, at a rate $\gamma\ ms^{-1}$. We have assumed that γ is the same for all split configurations, but this could be easily adapted, if we wanted to capture situations where some splits could last longer than others. Once the CU is at stand-by, frames stop being served. If there was any frame being served, it is kept at the cache of the processor (i.e. it does not occupy a buffer position), but its processing is frozen, until the stand-by situation finishes. As can be seen on the lower row in Figure 1, during the stand-by situation state transitions just happen rightwards, whenever a new frame arrives, but no frames are processed (i.e. they do not leave the system) and there do not exist leftwards state transitions.

We also model the time the controller stays at the stand-by configuration with an exponential random variable, with mean value $\frac{1}{\xi}\ ms$. Hence the rate at which the model leaves any state $(i, 0)$ equals $\xi\ ms^{-1}$. As can be observed, we consider that the next configuration split is randomly selected, with a certain probability. Thus, whenever the controller ends its stand-by configuration, it starts split k^{th} with probability α_k, with $\sum_{k=1\dots s} \alpha_k = 1$. This is reflected in Figure 1 with transition rates of $\alpha_k \cdot \xi$ from state $(i, 0)$ to (i, k), $\forall k = 1 \dots s$.

As can be seen, the defined model corresponds to a QBD process, which is a generalization of traditional birth-and-death process, where states are grouped into levels, in our case according to the number of frames at the controller (i.e.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

each column in Figure 1). The transitions only happen between states within the same level (change of split configuration) or to states in adjacent levels (upon a frame arrival or exit). As will be seen later, this yields to a block tridiagonal matrix representation of the underlying process. The reader can refer to the seminal works of Neuts [48] and Hajek [49], or complete books by Neuts himself [50], or Latouche and Ramaswami [51], for a more thorough discussion of the corresponding theoretical framework, in particular the so-called Matrix Geometric Method, which we will use in the next Section to analyze the behavior of the CU.

IV. Controller Performance

We resolve the model introduced above, using the Matrix Geometric Method, to find its stationary distribution and its average performance, in terms of time spent at the controller and loss probability. We first consider a controller with an infinite-capacity buffer, where frames are never lost (provided it works at a stable regime). Then we also study the performance of a limited-buffer controller.

A. Infinite buffer controller

By considering the transition rates of the Markov Chain in Figure 1 (without considering the last column at its right side, $\beta+1, k$), we can define the infinitesimal matrix of the corresponding QBD process:

$$\mathcal{Q} = \begin{bmatrix} L_0 & F & 0 & 0 & \cdots \\ B & L & F & 0 & \cdots \\ 0 & B & L & F & \cdots \\ \vdots & & \ddots & & \ddots \end{bmatrix} \tag{1}$$

where L_0, B, L, F are $(s+1)\times(s+1)$ matrices. B, F, L are given in equation (2). As can be seen, F is a diagonal matrix, with all its elements being λ (it thus corresponds to the identity matrix of the appropriate dimension multiplied by λ). B is also a diagonal matrix, with the first element of the diagonal being 0, while the rest correspond to the service rates of the various splits: $\mu_1 \dots \mu_s$. On the other hand, L_0 can be straightforwardly calculated as $L_0 = L + B$.

We then define the stationary distribution of the system as $\mathbf{\Pi} = [\pi_0, \pi_1 \dots]$, where π_i is a column vector of length $s+1$, so that $\pi_i(j)$ corresponds to the probability of having i frames in the CU, configured in the j^{th} split. Hence, $\pi_i(0)$ would be the probability of having i frames in the controller, while being on stand-by.

If the system is working in a stable regime, then there exists such stationary distribution, and it can be demonstrated that there is a constant matrix R so that [50, Theorem 3.1.1]:

$$R^2 \cdot B + R \cdot L + F = 0 \tag{3}$$

In addition, there is a unique positive solution to the following system of equations:

$$\begin{aligned} \pi_0^\intercal \left(L_0 + RB\right) &= \mathbf{0}^\intercal \\ \pi_0^\intercal \left(I - R\right)^{-1} \mathbf{1} &= 1 \end{aligned} \tag{4}$$

where $\mathbf{0}$ and $\mathbf{1}$ are column vectors, of length $s+1$, with all their elements being 0 and 1, respectively.

Then, $\mathbf{\Pi}$ is given by:

$$\pi_i^\intercal = \pi_0^\intercal \cdot R^i \tag{5}$$

Since there does not exist a closed solution for the quadratic equation in (3), an iterative method can be used to find R.[2]

We could also find the probability that the controller stays at a particular split configuration (or in stand-by), by summing the corresponding probabilities:

$$\mathcal{P}_{\text{split } k} = \sum_{i=0}^{\infty} \pi_i(k) \tag{6}$$

Such probability equals the fraction of time the controller would stay at a particular configuration. We could also establish it by means of the corresponding rates (ξ and γ), and the probabilities of selecting each split configuration α_k:

$$\mathcal{P}_{\text{split } k} = \begin{cases} \frac{\alpha_k \cdot \xi}{\gamma+\xi} & k = 1 \dots s \\ \frac{\gamma}{\gamma+\xi} & k = 0 \text{ (stand-by)} \end{cases} \tag{7}$$

We will afterwards use (7) to validate the model, by comparing it with the results yielded by (6).

The stationary distribution would allow us to establish the performance of the CU. Based on $\mathbf{\Pi}$, we can easily obtain the average number of frames in the controller:

$$\overline{n} = \sum_{i=0}^{\infty} i \cdot \|\pi_i\|_1 = \left\| \frac{\pi_0^\intercal \cdot R}{(I-R)^2} \right\|_1 \tag{8}$$

We can then apply Little's Law to find the average time a frame stays in the controller, $\overline{t_t}$, both waiting and at the processor. In this case, frame arrival rate is constant, λ, and so:

$$\overline{t_t} = \frac{\overline{n}}{\lambda} \tag{9}$$

All the previous results for the infinite-buffer CU controller are valid provided the system is operating at stable regime. In this sense, it is worth recalling that the corresponding QBD process has a stationary solution if and only if the following condition holds:

$$\eta B \mathbf{1} > \eta F \mathbf{1} \tag{10}$$

where η is the stationary probability vector (row vector) of matrix $A = B + L + F$, and $\mathbf{1}$ is a column vector, of the appropriate length $(s+1)$, with all its elements being 1.

By solving $\eta A = 0$, we obtain that: $\eta_0 = \frac{\gamma}{\gamma+\xi}$ and $\eta_k = \frac{\alpha_k \xi}{\gamma+\xi}, \forall k = 1 \dots s$, i.e. they match the split probabilities (7). Then, by substituting into (10), we can finally establish the stability condition for the CU operation:

$$\lambda < \frac{\gamma + \xi \sum_{k=1}^{s} \alpha_k \cdot \mu_k}{\gamma + \xi} \tag{11}$$

[2] In Matlab 2018, assuming a four-split controller, and an error of $\epsilon = 10^{-8}$, it takes approximately 230 iterations to find R, in less than 5 *ms*, using a laptop equipped with an i7 Intel processor.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

$$F = \begin{bmatrix} \lambda & 0 & \cdots & 0 \\ 0 & \lambda & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & \mu_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mu_s \end{bmatrix} \quad L = \begin{bmatrix} -(\lambda+\xi) & \alpha_1\xi & \cdots & \alpha_s\xi \\ \gamma & -(\lambda+\gamma+\mu_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \gamma & 0 & \cdots & -(\lambda+\gamma+\mu_s) \end{bmatrix} \tag{2}$$

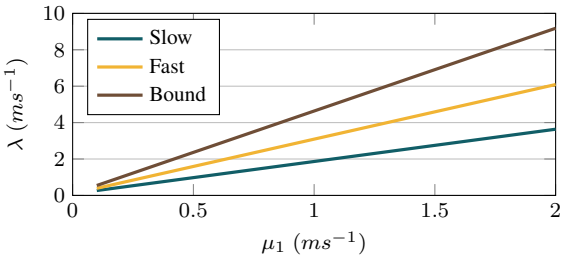


Fig. 2: Comparison of the λ bound to ensure stability conditions for fast and slow CU configurations

We can define a straightforward bound for λ, since in some cases, it might not be possible knowing the actual split probabilities. Since $\sum_{k=1}^{s} \alpha_k \cdot \mu_k \leq \max_k \mu_k$, we can conclude that:

$$\lambda < \frac{\gamma + \xi \left\|\mu\right\|_\infty}{\gamma + \xi} \tag{12}$$

In order to assess the looseness of the previous bound, we have made a simple experiment, using the CU configurations that will be used afterwards, in Section V. We assume that $\lambda = \xi = 1\ ms^{-1}$, and $\gamma = \frac{1}{10}\ ms^{-1}$. We consider a controller with four different splits, with service rates given by: $\mu = [1\ 1.5\ 3\ 5]$. Furthermore, we use two α vectors, so that both the fastest and slowest rates have a larger probability, leading to fast and slow setups. We then increase the value of μ_1, so augmenting the processing capacity of the controller, while keeping the ratio of the other service rates, i.e. $\mu^\dagger = \mu_1 \cdot [1\ 1.5\ 3\ 5]$. Results are shown in Figure 2. As can be seen, when μ_1 is large, the bound behaves worse, but for lower service rates, the bound provides a rather good approximation, in particular for the fast setup, to find the maximum arrival rate that could be accepted to guarantee the controller works within its stable regime.

B. Finite buffer controller

We assume in this case that the buffer has a finite capacity, to keep up to β frames before they can be eventually processed, which corresponds to the Markov Chain shown in Figure 1. The infinitesimal generator matrix of the QBD process, $\mathcal{Q}_b$ is the following finite matrix:

$$\mathcal{Q}_b = \begin{bmatrix} L_0 & F & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ B & L & F & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & B & L & F & \cdots & 0 & 0 & 0 & 0 \\ \vdots & & \ddots & & \ddots & & \ddots & & \vdots \\ 0 & 0 & 0 & 0 & \cdots & B & L & F & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & B & L & F \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & B & L_{\beta+1} \end{bmatrix} \tag{13}$$

all $L_0, B, L, F, L_{\beta+1}$ are $(s+1) \times (s+1)$ matrices. Hence, the dimension of $\mathcal{Q}_b$ equals $(s+1)\cdot(\beta+2) \times (s+1)\cdot(\beta+2)$. B, F, L are the same matrices that were seen for the infinite buffer model, given in (2). L_0 is still $L_0 = L + B$, while $L_{\beta+1} = L + F$.

In order to find the corresponding stationary distribution, we first establish the constant matrices V and $\widetilde{V}$, which are defined as [49, Theorem 1,§4]:

$$\begin{aligned} V &= -V \cdot F \cdot V \cdot B \cdot L^{-1} - L^{-1} \\ \widetilde{V} &= -\widetilde{V} \cdot B \cdot \widetilde{V} \cdot F \cdot L^{-1} - L^{-1} \end{aligned} \tag{14}$$

As was the case with R in (3), we use an iterative method to obtain V and $\widetilde{V}$[3]. Based on such matrices, we define $H, \widetilde{H}$, as follows [49, §4]:

$$H = F \cdot V \quad ; \quad \widetilde{H} = B \cdot \widetilde{V} \tag{15}$$

We can find the first and last rows of the fundamental matrix N of the corresponding process as can be seen in (16) [49, Theorem 4].

Afterwards, based on (16), we can finally build the matrix M [49, Theorem 5]:

$$M = \begin{bmatrix} L_0 + F \cdot N_{1,1} \cdot B & F \cdot N_{1,\beta} \cdot F \\ B \cdot N_{\beta,1} \cdot B & L_{\beta+1} + B \cdot N_{\beta,\beta} \cdot F \end{bmatrix} \tag{17}$$

We can then establish $\mathbf{x}_0$ and $\mathbf{x}_{\beta+1}$ as the invariant probability distribution of M (its left eigenvector, having an eigenvalue of 1): $\left[\mathbf{x}_0^\intercal\ \mathbf{x}_{\beta+1}^\intercal\right]$. We can finally write that [49, Theorem 5]:

$$\mathbf{x}_k^\intercal = \mathbf{x}_0^\intercal \cdot F \cdot N_{1,k} + \mathbf{x}_{\beta+1}^\intercal \cdot B \cdot N_{\beta,k} \quad k = 1 \ldots \beta \tag{18}$$

Finally, the probability distribution of the corresponding Markov Chain states is defined as:

$$\mathbf{\Pi}_b = \frac{1}{c} \left[\mathbf{x}_0\ \mathbf{x}_1 \ldots \mathbf{x}_{\beta+1}\right] \tag{19}$$

[3]In this case, the number of iterations that were needed for a controller with four splits, and an error of $\epsilon = 10^{-8}$ were, approximately, 220 and Matlab 2018 took, in a laptop with an i7 Intel processor, ≈ 10 *ms*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

$$\begin{bmatrix} N_{1,1} & \cdots & N_{1,\beta} \\ N_{\beta,1} & \cdots & N_{\beta,\beta} \end{bmatrix} = -\begin{bmatrix} L + H \cdot B & -H^{\beta} \cdot B \\ -\widetilde{H}^{\beta} \cdot F & \widetilde{H} \cdot F + L \end{bmatrix}^{-1} \cdot \begin{bmatrix} I & H & \cdots & H^{\beta-2} & H^{\beta-1} \\ \widetilde{H}^{\beta-1} & \widetilde{H}^{\beta-2} & \cdots & \widetilde{H} & I \end{bmatrix} \tag{16}$$

where c is a normalizing constant, which guarantees that the sum of all probabilities equals 1:

$$c = \left(\sum_{k=0}^{\beta+1} \|\mathbf{x}_k\|_1 \right)^{-1} \tag{20}$$

From the stationary state distribution, we can (as was done for the infinite buffer model) characterize the performance of the system. We can first calculate the probability of working at a particular split, which should also equal the results that were shown in (7):

$$\mathcal{P}_{\text{split } k} = \sum_{i=0}^{\beta+1} \pi_i(k) \tag{21}$$

We can also find the average number of frames in the system, as:

$$\overline{n} = \sum_{i=0}^{\beta+1} i \cdot \|\pi_i\|_1 \tag{22}$$

In order to find the average time at the controller using Little's Law, we first need to calculate the average served frame rate (since in this case there are losses). $\overline{\lambda}$ can be calculated as:

$$\overline{\lambda} = \sum_{k=0}^{\beta} \lambda \cdot \|\pi_k\|_1 \tag{23}$$

We can finally write that:

$$\overline{t_t} = \frac{\overline{n}}{\overline{\lambda}} \tag{24}$$

As was said earlier, the main difference between the controller with infinite buffer and this one is that frames are lost, when the buffer is fully occupied, and a frame arrives. Exploiting the Poisson-arrivals-see-time-averages (PASTA) property [52], we can obtain the loss probability as follows:

$$\mathcal{P}_{\text{loss}} = \|\pi_{\beta+1}\|_1 \tag{25}$$

In the next section we will discuss all the results that can be obtained by using the presented model, validating it by means of extensive simulations.

V. Results

In this section, we discuss the results yielded by the proposed model. We compare them with the performance obtained by means of an extensive experiment campaign that was carried out using a proprietary event-driven simulator, implemented in *C++*. This allows us to assess the validity of both the model and the simulator, which will be afterwards used to broaden the analysis.

TABLE II: Configuration of analyzed scenarios

Scenario	#Splits	α	μ	γ
$\mathcal{A}$	4	$[0.4, 0.3, 0.2, 0.1]$	$[1, 1.5, 3, 5]$	0.1
$\mathcal{B}$	4	$[0.1, 0.2, 0.3, 0.4]$	$[1, 1.5, 3, 5]$	0.1

The simulator defines four different types of events: (1) arrival of a new frame; (2) frame service completion (a frame exits the CU); (3) split configuration change (transition to stand-by); (4) end of the stand-by state. The events are processed according to the CU state, which takes into account the current number of frames. For instance, whenever a frame arrival event is processed, the simulator engine checks whether the processor is empty and, if such is the case, it schedules the exit of such frame, using an instance of the corresponding time distribution. On the other hand, if there was already another frame being processed, the buffer length is increased. As can be seen, differently to the proposed model, the simulator is not constrained by any particular time distribution, so that different choices can be used to schedule the forthcoming events, according to the particular experiment configurations. In this sense, this tool can be also used to analyze the behavior when the model assumptions do not hold[4].

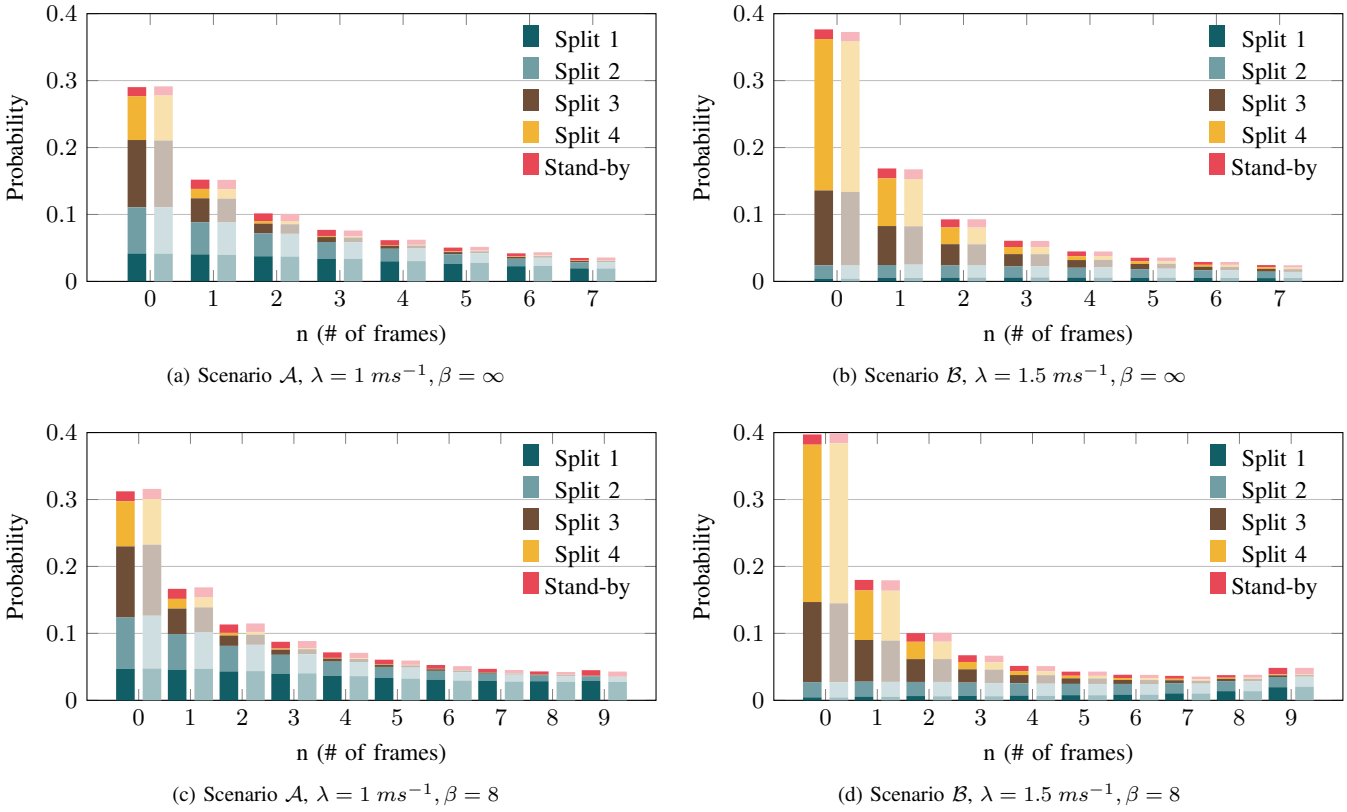
We consider the scenarios that are depicted in Table II. We assume that the CU has four different splits, each of them with a different service rate, sorted in ascending order. In the first scenario ($\mathcal{A}$), the slowest service is more likely to be used, while in the second setup ($\mathcal{B}$), the fastest service rate has a greater probability. We further assume that $\gamma = 0.1\ ms^{-1}$ and $\xi = 1\ ms^{-1}$. For the finite buffer controller, we will study the impact of its length, while we will ensure that the infinite buffer CU always operates at a stable regime.

We start by analyzing the probability of having a number of frames in the controller. Figure 3 shows such probabilities, for the two aforementioned scenarios, and for both controller models. Upper subfigures correspond to the infinitive buffer ($\beta = \infty$) controller, while the bottom plots show the results for the finite buffer, for which we used a buffer length of $\beta = 8$. For scenario $\mathcal{A}$ we assume a frame rate of $\lambda = 1\ ms^{-1}$, and we increase it to $\lambda = 1.5\ ms^{-1}$ for setup $\mathcal{B}$. In both cases, we guarantee that the infinite-buffer controller is working at a stable regime. With the simulator, we run experiments comprising 200000 frames, to ensure statistically tight results.

We use a stacked bar representation, so that each bar corresponds to a particular value of frames, as indicated in the x-axis (n), and it is as well divided into a number of chunks, which reflect the probability of each of the four splits and the stand-by situation. First of all, we can see that there is an almost perfect match between the theoretical results and those

[4]More details are given in the README file in https://github.com/ldiez/5GvRanControllerQBD

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management



(a) Scenario $\mathcal{A}$, $\lambda = 1\ ms^{-1}, \beta = \infty$

(b) Scenario $\mathcal{B}$, $\lambda = 1.5\ ms^{-1}, \beta = \infty$

(c) Scenario $\mathcal{A}$, $\lambda = 1\ ms^{-1}, \beta = 8$

(d) Scenario $\mathcal{B}$, $\lambda = 1.5\ ms^{-1}, \beta = 8$

Fig. 3: Probability of having n packets in the controller. Model and simulation results are shown with solid and shaded colors respectively

obtained with the simulator, which proves their correctness. We can also see that the presence of the faster splits is much more relevant for low n values, and they quickly become less visible when n gets higher. This is sensible, since the controller would be able to serve more frames when working at a quicker pace, and it would thus take less time to reach the corresponding $(0, k)$ state. On the other hand, the stand-by probability does not vary with n, as could have been expected, since the values of ξ and γ do not depend on the service rates, nor on the particular configuration.

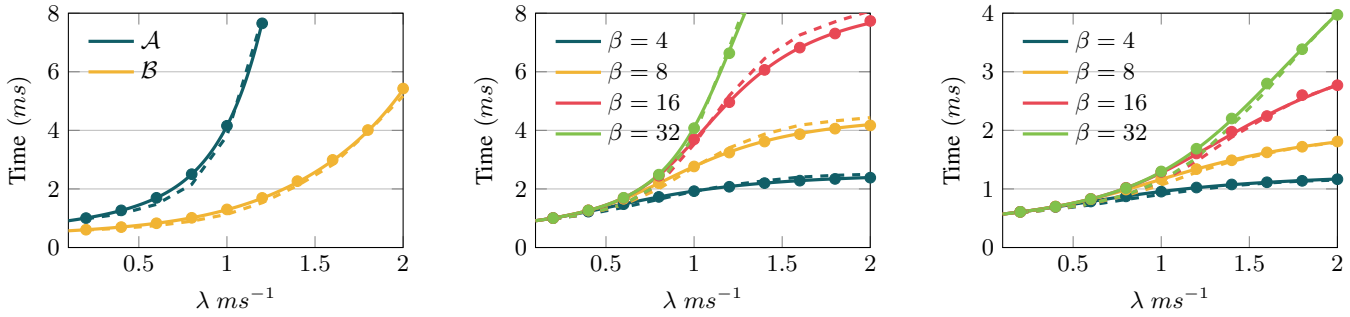
The aforementioned match between analytical and simulation results is also reflected in the probabilities of working at a particular split. We recall that these are given by (7). Table III shows the corresponding probabilities for the four configurations, and the values that were obtained with both the proposed model and the simulator. As can be seen, the values that are obtained by summing the corresponding state probabilities, as given by (6) and (21), perfectly match the theoretical values (7). On the other hand, the results obtained with the simulator (again after a single experiment comprising 200000 frames) are also very close.

We now analyze the performance of the CU, considering both the time that a frame stays at the controller, as well as the loss probability (for the finite-buffer case). Figure 4 shows the average time a frame stays at the controller. In all cases, we represent with solid lines the theoretical results (those obtained by exploiting the model discussed in Section III), and

TABLE III: Split probabilities

Scen.	Buffer	Split	Theor. (7)	Model (6), (21)	Simul.
$\mathcal{A}$ λ=1 ms^{-1}	$\beta = \infty$	Split 1	0.3636	0.3636	0.3593
		Split 2	0.2727	0.2727	0.2764
		Split 3	0.1818	0.1818	0.1801
		Split 4	0.0909	0.0909	0.0934
		Stand-by	0.0909	0.0909	0.0907
	$\beta = 8$	Split 1	0.3636	0.3636	0.3604
		Split 2	0.2727	0.2727	0.2761
		Split 3	0.1818	0.1818	0.1813
		Split 4	0.0909	0.0909	0.0922
		Stand-by	0.0909	0.0909	0.0900
$\mathcal{B}$ λ=1.5 ms^{-1}	$\beta = \infty$	Split 1	0.0909	0.0909	0.0916
		Split 2	0.1818	0.1818	0.1852
		Split 3	0.2727	0.2727	0.2698
		Split 4	0.3636	0.3636	0.3622
		Stand-by	0.0909	0.0909	0.0912
	$\beta = 8$	Split 1	0.0909	0.0909	0.0893
		Split 2	0.1818	0.1818	0.1809
		Split 3	0.2727	0.2727	0.2714
		Split 4	0.3636	0.3636	0.3681
		Stand-by	0.0909	0.3636	0.0903

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management



(a) Scenarios $\mathcal{A}$ and $\mathcal{B}$, $\beta = \infty$ (b) Scenario $\mathcal{A}$ with finite buffer (c) Scenario $\mathcal{B}$ with finite buffer

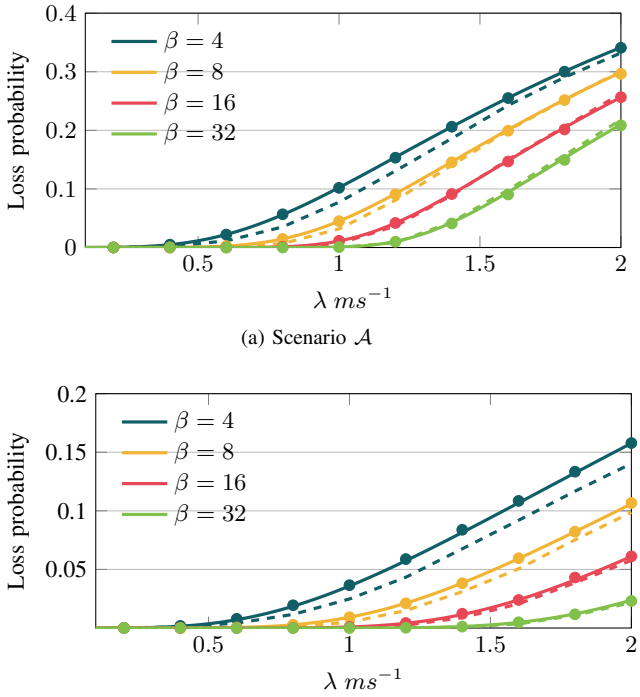
Fig. 4: Total time Vs. λ. Model results are shown with solid lines. Simulation results with exponential and constant service time are represented with markers and dashed lines, respectively

markers correspond to the values yielded by the simulation, after 100 independent experiments, each of them comprising 10000 frames, ensuring statistical tightness. Last, dashed lines are the results that were obtained by using the simulator, using constant service times, instead of exponentially distributed ones.

First of all we can again see the almost perfect match between the theoretical results and the values offered by the simulator. More interestingly, the difference when constant service times are used is not very relevant. In the case of the infinite buffer controller (Figure 4a) we can observe that the average time in the controller heavily increases with λ. This is more relevant for scenario $\mathcal{A}$, where the λ range that guarantees a stable regime is shorter. The behavior is different when a finite buffer is used, as can be seen in Figures 4b and 4c, for setups $\mathcal{A}$ and $\mathcal{B}$, respectively. We can see that the time tends to stabilize when λ gets higher. In addition, it also increases as long as the capacity of the buffer is larger. Both observations could have been expected. When the buffer capacity is lower, there would be more losses (this will be discussed afterwards) for greater arrival rates, but the time a frame waits to be served does not increase, and is limited. In this sense, we can see that the aforementioned stabilization happens earlier when the buffer length is shorter, while we do not even see this for the longest buffers ($\beta = 32$). For the finite-buffer controller we can also see that there is a slightly larger difference between the constant and exponentially distributed service times, as long as b increases, being more relevant in scenario $\mathcal{A}$, which is characterized by a larger load.

For the two types of controller, these results would yield a reasonable limit on the maximum allowable frame arrival rate. As was mentioned previously, 5G use cases set stringent delay requirements, and the presented model can shed light on the impact that CU might have. In the case of the finite-buffer, we can see that such time does not heavily increase for short buffer lengths, although as will be seen below, there is a strong impact on the corresponding frame loss probability.

We have assumed that the infinite buffer controller, provided that the system works in a stable regime, would keep frames in the buffer until they can be eventually served, and so there do not exist losses. In more realistic scenarios, it is sensible to assume that the CU would actually have a certain buffer



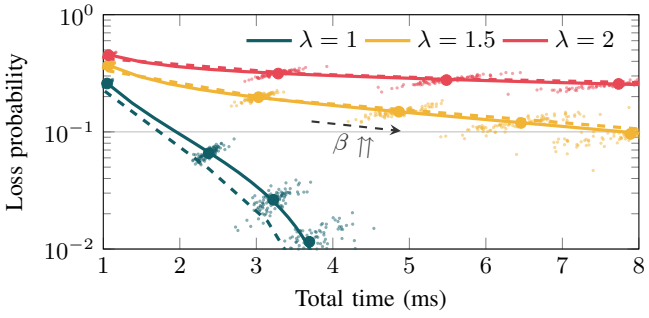
(a) Scenario $\mathcal{A}$

(b) Scenario $\mathcal{B}$

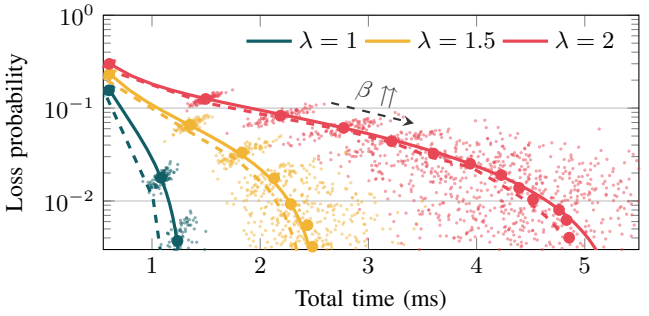
Fig. 5: Loss probability Vs. λ. Model and simulation results are shown with solid lines and markers, respectively. Dashed lines correspond to constant service rates

length, and incoming frames would be lost if there is not available capacity. Figure 5 shows the loss probability that was obtained for the finite-buffer CU, for the two setups. We keep using solid lines to represent the results yielded by the proposed model, markers correspond to the values obtained with the simulator, while the dashed lines are the loss probabilities that were seen (again exploiting the simulator) when using constant service times, instead of exponentially distributed. All the results obtained with the simulator are based on 100 independent experiments, each of them entailing the transmission of 10000 frames. There is again a perfect match between theoretical and simulation results. As was expected, loss probability increases with λ, and it is more relevant for shorter buffer lengths. It is worth noting that,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

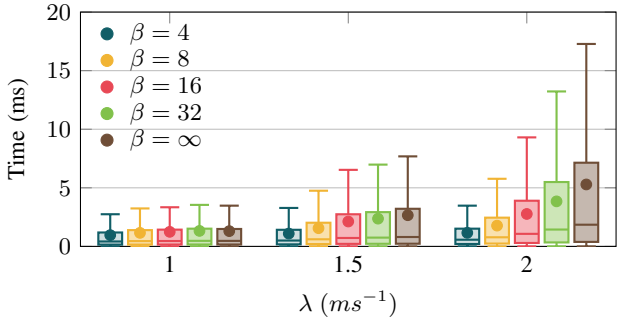


(a) Scenario $\mathcal{A}$

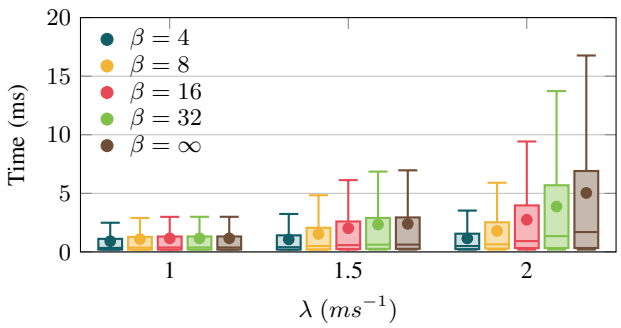


(b) Scenario $\mathcal{B}$

Fig. 6: Loss probability Vs. transfer time, while increasing the buffer length (b). Model results are shown with solid lines. Simulation results with exponential and constant service time are represented with markers and dashed lines, respectively. Point cloud represents the sparsity of the simulation results for exponential service time



(a) Exponential service time



(b) Constant service time

Fig. 7: Whisker plot of simulated total time Vs. λ. Scenario $\mathcal{B}$ for exponential and constant service time, and different buffer size

despite the finite-buffer model does not strictly impose a maximum frame arrival rate, the loss probabilities that are seen for large λ are clearly unacceptable. Again, the proposed model would allow addressing the design of these systems, based on the requirements of the corresponding services.

Opposed to the average time (see Figures 4), the differences observed between the loss probabilities when constant service times are used, instead of the exponentially distributed ones, are more relevant as long as the buffer length gets shorter.

The results discussed so far for the finite-buffer controller have shown that longer buffer lengths would yield lower loss probabilities, but at the cost of heavily increasing the time spent at the controller. This sets a clear trade-off between these two performance indicators, since in 5G loss probabilities should be kept low, but the average transfer time should not heavily increase if the stringent delay requirements are to be respected. In order to study this trade-off, Figure 6 shows the performance region for the finite-buffer controller. We fix λ to three different values, and we increase the buffer length. The figure plots how the loss probability varies against the average total time in the controller. As in previous results, solid lines were yielded by the proposed model, markers correspond to the results obtained with the simulator (100 independent runs with 10000 frames each), while the dashed lines show the values (again using the simulator) that were obtained for constant service times. In this case we also represent, as point clouds, all the values that were obtained with the simulator, to show the variability of the results. We can still see a very tight match between the values yielded by the proposed model, and the simulator results (with a slight difference for high λ and large buffers (see the red line in Figure 6b). The differences that were observed in Figure 5 between constant and exponentially distributed service times are reflected as well.

For scenario $\mathcal{A}$, which has a larger load (slower CU configuration), we can see that, when λ is high, increasing the buffer does not heavily improve the performance of the controller, with a slight decrease of loss probability, albeit relevant longer transfer times. However, in the results obtained for scenario $\mathcal{B}$, we can observe how the increase of the buffer length might yield a sharp decrease on the loss probability without a relevant increase of the time spent at the controller. We can also see that for lower λ and shorter buffer lengths, the performance of the controller is much more predictable, and the corresponding results are rather tight, both in terms of delay and loss probability. However, when the buffer length increases, there is a much wider range of performances, although the average behavior matches the one yielded by the proposed model. This is corroborated by the next set of results.

So far, we have focused on the average behavior exhibited by the controller, but it is also interesting to assess the variability of such performance indicators. In order to study their statistical tightness, we will exploit the simulator, since the proposed theoretical model only yields the average values.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management

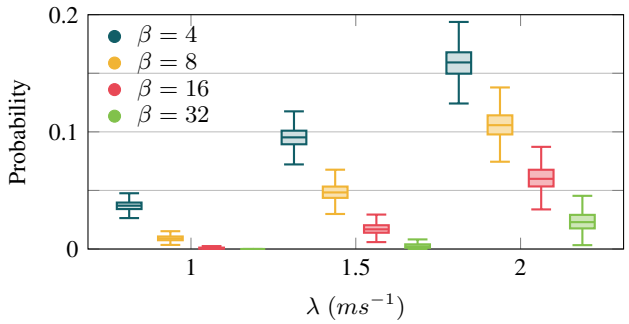
First, Figure 7 uses whisker plots to capture the variability of the time frames spend at the controller, both for the infinite-buffer ($\beta = \infty$) and finite-buffer (with various buffer lengths) CU. Each whisker plot corresponds to a single experiment, comprising 200000 frames, and it shows the median (0.5 percentile) of the time (horizontal line within the box), the 0.25 and 0.75 percentiles (upper and lower box limits), as well as the 0.05 and 0.95 percentiles. We have also included (with a circular marker) the average value, which corresponds to the ones shown in the previous figures. The upper figure corresponds to the exponentially distributed time, while the lower one shows the results obtained when the service time was constant.

As can be seen, for low λ, the variability is rather low, and it does not depend on the buffer length (it does not even increase for the infinite-buffer configuration). However, when the frame arrival rate increases, we can see that the variability of the time frames need to stay at the controller gets considerably higher. This is more relevant as long as the length of the buffer is larger. In fact, the highest variance is seen for the infinite-buffer model.

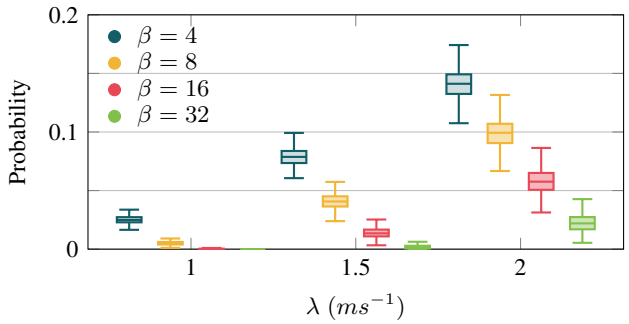
In addition, Figure 8 shows the variability of the loss probability. The configurations are the same that were just described, when discussing Figure 7. In this case we do not include the infinite-buffer controller, since frames are not lost. Again we see that the larger λ, the larger the variability, although the increase is not as relevant as it was seen for the transfer time (this was also observed in Figure 6). Furthermore, while we saw that the variance exhibited by the time was strongly affected by the corresponding buffer length, this is not reflected in the loss probability, and different buffer lengths yield similar variabilities for the same arrival rate.

We have also exploited the simulator to analyze the behavior of a realistic controller configurations. We have used the metrics reported in [53], where the authors assess the performance of real dynamic functional split implementations. In particular, they study the standby time under different split changing implementations. As it is our case, the authors do not propose particular split selection policies, but they implement a SDR based solution able to be used by a given policy. We used the values shown in [53, Fig. 5]. In particular we configured the simulator with a controller having 2 split options, which correspond to the MAC-PHY and PDCP-RLC splits implemented in [53]. Traffic arrives with an Inter arrival Time (IaT) between frames of 8 ms (i.e. yielding a frame arrival rate $\lambda = 0.125\ ms^{-1}$). According to the capacity values mentioned in [53, Fig. 4], service times are set to 5 and 10 ms for each split respectively. Hence, service rates are $\mu_1 = 0.2\ ms^{-1}$ and $\mu_2 = 0.1\ ms^{-1}$. Eventually, the standby time is modeled using a uniform distribution, as shown in [53, Fig. 5], as well as using an exponential distribution, to assess the impact of of the assumptions the model takes. We consider that both split configurations are equiprobable, i.e. $\alpha_1 = \alpha_2 = 0.5$, and the average time between split changes is 10 times higher than the standby duration.

Figure 9 compares the controller performance when both uniform and exponential distributions are assumed for the standby time, using different buffer sizes. First, Figures 9a and 9b show the distribution of the time frames spend at the controller, and the loss probability, respectively. We can see that the behavior is similar than the one observed earlier. When we increase the buffer length, we can notably reduce the loss probability, at the cost of heavily increasing the time spent at the controller. We can also see that the impact of assuming the exponential distribution for the stand-by duration is not very relevant, since the results obtained with the uniform distribution are rather similar, both in terms of the average value and the corresponding variability. Finally, Figure 9c shows the trade-off between loss probability and total time at the controller, as we increase the buffer length. Once again, the results evince that both configurations yield very similar behaviors. As was also seen earlier, greater buffer lengths lead to smaller loss probabilities, at the cost of longer delays, which might hinder the strict delay requirements of 5G communications. We can also see that such configurations are also characterized by showing a less predictable behavior, with a larger variability in the results.



(a) Exponential service time



(b) Constant service time

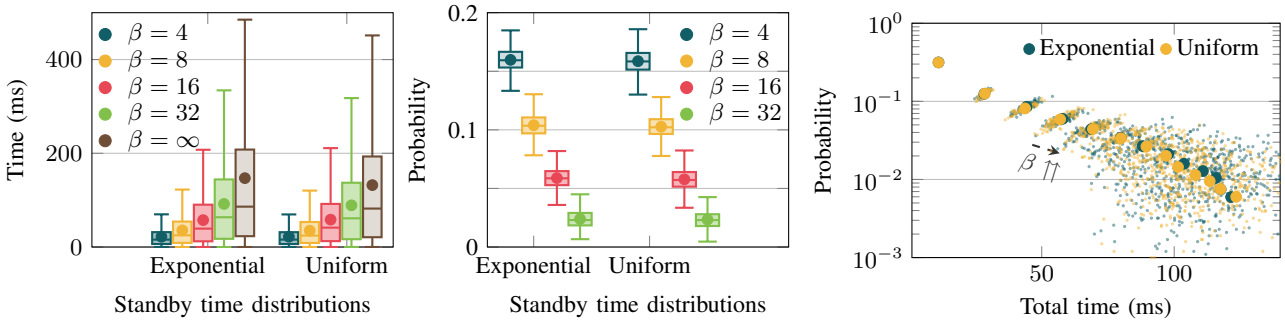
Fig. 8: Whisker plot of simulated loss probability Vs. λ. Scenario $\mathcal{B}$ for exponential and constant service time, and different buffer size

VI. Conclusion

We have introduced a novel model that can be used to appropriately capture the performance of Flexible Functional Split in vRAN 5G controllers. In spite of the growing interest on this type of network elements, and to our best knowledge, this is the first proposal that can be effectively used to tackle their adequate dimensioning.

The model is based on a bi-dimensional Markov Chain, which can consider both infinite and finite-buffer controllers.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TNSM.2020.3045968, IEEE Transactions on Network and Service Management



(a) Total time at the controller

(b) Loss probability

(c) Loss probability Vs. Total time at the controller, increasing buffer length

Fig. 9: Realistic controller (configuration taken from [53]) performance for exponential and uniform distributions for the standby duration

In the former group, we assume that the controller has enough capacity in its buffer to keep incoming frames, until they can be eventually served, and so there do not exist any losses. In the second group, we assume the controller to have a limited buffer, and so frames would be lost. Matrix geometric techniques have been use to solve the corresponding Quasi-Birth-Death process.

A proprietary event-driven simulator has been first used to assess the validity of the proposed model. The obtained results, after an extensive simulation campaign, show an almost perfect match between both approaches, which are thus validated. Then, we have exploited the simulator to broaden the analysis, considering different service time distributions, as well as the statistical variability of the performance indicators.

For the infinite-buffer model we have focused on the time frames need to stay at the controller. Based on 5G stringent delay requirement, this could be used to establish sensible configurations, or a maximum admissible frame arrival rate. On the other hand, for the finite-buffer controller, we have studied the trade-off between transfer time and loss probability, and the impact of the buffer length. We also used realistic configuration values of a real controller implementation to assess its peformance, by exploiting the developed simulator.

The proposed model can undoubtedly serve to tackle an appropriate design of vRAN controllers, since it can shed light on the impact of their configuration over performance indicators, to ensure that requirements posed by 5G services are adequately met. In this sense, we have made the code available to the scientific community, in a public github repository.

In our future work, we plan to exploit the model in more specific scenarios and use-cases, making the corresponding adjustments in the controller configuration. We are also starting to look at the performance of these network elements on a lower scale, by considering scheduling policies and dynamic queue management techniques. In this sense, we will use both the model and the simulator to evakuate the behavior of particular split selection policies, which might yield optimum performance in terms of a variety of metrics, including latency and loss probability.

ACKNOWLEDGMENT

This work has been funded by the Spanish Government (Ministerio de Economía y Competitividad, Fondo Europeo de Desarrollo Regional, MINECO-FEDER) by means of the project FIERCE: Future Internet Enabled Resilient smart CitiEs (RTI2018-093475-AI00).

REFERENCES

[1] M. Peng, Y. Sun, X. Li, Z. Mao, and C. Wang, "Recent Advances in Cloud Radio Access Networks: System Architectures, Key Techniques, and Open Issues," *IEEE Communications Surveys Tutorials*, vol. 18, no. 3, pp. 2282–2308, 2016.

[2] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Communications Surveys Tutorials*, vol. 21, no. 1, pp. 146–172, Firstquarter 2019.

[3] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks—A Technology Overview," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 405–426, Firstquarter 2015.

[4] J. Wu, Z. Zhang, Y. Hong, and Y. Wen, "Cloud radio access network (C-RAN): a primer," *IEEE Network*, vol. 29, no. 1, pp. 35–41, Jan 2015.

[5] G. O. Pérez, J. A. Hernández, and D. Larrabeiti, "Fronthaul network modeling and dimensioning meeting ultra-low latency requirements for 5G," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 6, pp. 573–581, June 2018.

[6] C. I, Y. Yuan, J. Huang, S. Ma, C. Cui, and R. Duan, "Rethink fronthaul for soft RAN," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 82–88, Sep. 2015.

[7] A. Checko, A. P. Avramova, M. S. Berger, and H. L. Christiansen, "Evaluating C-RAN fronthaul functional splits in terms of network level energy and cost savings," *Journal of Communications and Networks*, vol. 18, no. 2, pp. 162–172, 2016.

[8] 3GPP, "Study on new radio access technology: Radio access architecture and interfaces," 3rd Generation Partnership Project (3GPP), TR 38.801, 2017. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3056

[9] N. J. Gomes, P. Sehier, H. Thomas, P. Chanclou, B. Li, D. Munch, P. Assimakopoulos, S. Dixit, and V. Jungnickel, "Boosting 5G Through Ethernet: How Evolved Fronthaul Can Take Next-Generation Mobile to the Next Level," *IEEE Vehicular Technology Magazine*, vol. 13, no. 1, pp. 74–84, 2018.

[10] T. Truong-Huu, P. Murali Mohan, and M. Gurusamy, "Service chain embedding for diversified 5g slices with virtual network function sharing," *IEEE Communications Letters*, vol. 23, no. 5, pp. 826–829, 2019.

[11] A. M. Alba, J. H. G. Velásquez, and W. Kellerer, "An adaptive functional split in 5G networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2019, pp. 410–416.

[12] D. Harutyunyan and R. Riggio, "Flex5G: Flexible Functional Split in 5G Networks," *IEEE Transactions on Network and Service Management*, vol. 15, no. 3, pp. 961–975, Sep. 2018.

[13] ——, "Flexible functional split in 5G networks," in *2017 13th International Conference on Network and Service Management (CNSM)*, Nov 2017, pp. 1–9.

[14] P. Arnold, N. Bayer, J. Belschner, and G. Zimmermann, "5G radio access network architecture based on flexible functional control / user plane splits," in *2017 European Conference on Networks and Communications (EuCNC)*, June 2017, pp. 1–5.

[15] C. Chang, N. Nikaein, and T. Spyropoulos, "Impact of Packetization and Scheduling on C-RAN Fronthaul Performance," in *2016 IEEE Global Communications Conference (GLOBECOM)*, 2016, pp. 1–7.

[16] C. Chang, R. Schiavi, N. Nikaein, T. Spyropoulos, and C. Bonnet, "Impact of packetization and functional split on C-RAN fronthaul performance," in *2016 IEEE International Conference on Communications (ICC)*, 2016, pp. 1–7.

[17] Y. Zhou, J. Li, Y. Shi, and V. W. S. Wong, "Flexible Functional Split Design for Downlink C-RAN With Capacity-Constrained Fronthaul," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6050–6063, 2019.

[18] C. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, "FlexCRAN: A flexible functional split framework over ethernet fronthaul in Cloud-RAN," in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.

[19] Y. Alfadhli, M. Xu, S. Liu, F. Lu, P. Peng, and G. Chang, "Real-Time Demonstration of Adaptive Functional Split in 5G Flexible Mobile Fronthaul Networks," in *2018 Optical Fiber Communications Conference and Exposition (OFC)*, March 2018, pp. 1–3.

[20] A. Martinez Alba and W. Kellerer, "A Dynamic Functional Split in 5G Radio Access Networks," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[21] A. Alabbasi, M. Berg, and C. Cavdar, "Delay Constrained Hybrid CRAN: A Functional Split Optimization Framework," in *2018 IEEE Globecom Workshops (GC Wkshps)*, 2018, pp. 1–7.

[22] A. Sriram, M. Masoudi, A. Alabbasi, and C. Cavdar, "Joint Functional Splitting and Content Placement for Green Hybrid CRAN," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1–7.

[23] L. Wang and S. Zhou, "Flexible Functional Split in C-RAN with Renewable Energy Powered Remote Radio Units," in *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2018, pp. 1–6.

[24] D. A. Temesgene, M. Miozzo, and P. Dini, "Dynamic Functional Split Selection in Energy Harvesting Virtual Small Cells Using Temporal Difference Learning," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1813–1819.

[25] L. Wang and S. Zhou, "Energy-Efficient UAV Deployment with Flexible Functional Split Selection," in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.

[26] ——, "Flexible functional split and power control for energy harvesting cloud radio access networks," *IEEE Transactions on Wireless Communications*, vol. 19, no. 3, pp. 1535–1548, 2020.

[27] H. Gupta, M. Sharma, A. Franklin A., and B. R. Tamma, "Apt-ran: A flexible split-based 5g ran to minimize energy consumption and handovers," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 473–487, 2020.

[28] A. Marotta, D. Cassioli, K. Kondepu, C. Antonelli, and L. Valcarenghi, "Exploiting split in converged software defined access networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 11, no. 11, pp. 536–546, 2019.

[29] ——, "Efficient Management of Flexible Functional Split through Software Defined 5G Converged Access," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[30] Y. Li, J. Mårtensson, M. Fiorani, B. Skubic, Z. Ghebretensaé, Y. Zhao, J. Zhang, L. Wosinska, and P. Monti, "Flexible RAN: A Radio Access Network Concept with Flexible Functional Splits and a Programmable Optical Transport," in *2017 European Conference on Optical Communication (ECOC)*, Sep. 2017, pp. 1–3.

[31] Y. Li, J. Martensson, B. Skubic, Y. Zhao, J. Zhang, L. Wosinska, and P. Monti, "Flexible ran: Combining dynamic baseband split selection and reconfigurable optical transport to optimize ran performance," *IEEE Network*, vol. 34, no. 4, pp. 180–187, 2020.

[32] K. Kondepu, A. Sgambelluri, N. Sambo, F. Giannone, P. Castoldi, and L. Valcarenghi, "Orchestrating lightpath recovery and flexible functional split to preserve virtualized ran connectivity," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 10, no. 11, pp. 843–851, 2018.

[33] B. Ojaghi, F. Adelantado, E. Kartsakli, A. Antonopoulos, and C. Verikoukis, "Sliced-RAN: Joint Slicing and Functional Split in Future 5G Radio Access Networks," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, 2019, pp. 1–6.

[34] O. I. Abdullaziz, M. Capitani, C. E. Casetti, C. F. Chiasserini, S. B. Chundrigar, G. Landi, X. Li, F. Moscatelli, K. Sakaguchi, and S. T. Talat, "Energy monitoring and management in 5G integrated fronthaul and backhaul," in *2017 European Conference on Networks and Communications (EuCNC)*, 2017, pp. 1–6.

[35] S. S. Tadesse, C. Casetti, C. F. Chiasserini, and G. Landi, "Energy-efficient traffic allocation in SDN-basec backhaul networks: Theory and implementation," in *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, 2017, pp. 209–215.

[36] X. Li, R. Ferdous, C. F. Chiasserini, C. E. Casetti, F. Moscatelli, G. Landi, R. Casellas, K. Sakaguchi, S. B. Chundrigar, R. Vilalta, J. Mangues, A. Garcia-Saavedra, X. Costa-Pérez, L. Goratti, and D. Siracusa, "Novel Resource and Energy Management for 5G integrated backhaul/fronthaul (5G-Crosshaul)," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 778–784.

[37] A. Garcia-Saavedra, J. X. Salvat, X. Li, and X. Costa-Perez, "WizHaul: On the Centralization Degree of Cloud RAN Next Generation Fronthaul," *IEEE Transactions on Mobile Computing*, vol. 17, no. 10, pp. 2452–2466, 2018.

[38] A. Garcia-Saavedra, G. Iosifidis, X. Costa-Perez, and D. J. Leith, "Joint Optimization of Edge Computing Architectures and Radio Access Networks," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 11, pp. 2433–2443, 2018.

[39] A. Alameer and A. Sezgin, "Optimization framework for baseband functionality splitting in C-RAN," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2017, pp. 1–5.

[40] I. Koutsopoulos, "Optimal functional split selection and scheduling policies in 5G Radio Access Networks," in *2017 IEEE International Conference on Communications Workshops (ICC Workshops)*, May 2017, pp. 993–998.

[41] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, "CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 7993–8006, 2018.

[42] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "VrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1–16. [Online]. Available: https://doi.org/10.1145/3300061.3345431

[43] P. Rost, A. Maeder, M. C. Valenti, and S. Talarico, "Computationally Aware Sum-Rate Optimal Scheduling for Centralized Radio Access Networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[44] K. Wang, K. Yang, and C. S. Magurawalage, "Joint Energy Minimization and Resource Allocation in C-RAN with Mobile Cloud," *IEEE Transactions on Cloud Computing*, vol. 6, no. 3, pp. 760–770, 2018.

[45] Z. Cheng, Y. Tang, and H. Wu, "Joint Task Offloading and Flexible Functional Split in 5G Radio Access Network," in *2019 International Conference on Information Networking (ICOIN)*, 2019, pp. 114–119.

[46] M. Shehata, A. Elbanna, F. Musumeci, and M. Tornatore, "Multiplexing Gain and Processing Savings of 5G Radio-Access-Network Functional Splits," *IEEE Transactions on Green Communications and Networking*, vol. 2, no. 4, pp. 982–991, 2018.

[47] A. Mathew, M. Srinivasan, and C. S. R. Murthy, "Packet Generation Schemes and Network Latency Implications in SDN-enabled 5G C-RANs: Queuing Model Based Analysis," in *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2019, pp. 1–7.

[48] M. Neuts, "Markov Chains with Applications in Queueing Theory, Which Have a Matrix-Geometric Invariant Probability Vector," *Advances in Applied Probability*, vol. 10, no. 1, pp. 185–212, 1978.

[49] B. Hajek, "Birth-and-death processes on the integers with phases and general boundaries," *Journal of Applied Probability*, vol. 19, no. 3, p. 488–499, 1982.

[50] M. Neuts, *Matrix-geometric Solutions in Stochastic Models: An Algorithmic Approach*. Johns Hopkins University Press, 1981.

[51] G. Latouche and V. Ramaswami, *Introduction to Matrix Analytic Methods in Stochastic Modeling*. Society for Industrial and Applied Mathematics, 1999.

[52] R. W. Wolff, "Poisson Arrivals See Time Averages," *Operations Research*, vol. 30, no. 2, pp. 223–231, 1982.

[53] A. Martinez-Alba, J. H. G. Velásquez, and W. Kellerer, "An adaptive functional split in 5g networks," in *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops*, 2019, pp. 410–416.

Luis Diez received his M.Sc. and Ph.D. from University of Cantabria in 2013 and 2018 respectively. He is currently Assistant Professor at the Communications Engineering Department in that University. He has been involved in different international and industrial research projects. His research focuses on future network architectures, resource management in wireless heterogeneous networks, and IoT solutions and services. He has published more than 40 scientific and technical papers in those areas, and he has served as TPC member and reviewer in a number of international conferences and journals. As for teaching, Dr. Diez has supervised 15 BSc and MSc Thesis, and he teaches in courses related to cellular networks, network dimensioning and service management.

Cristina Hervella received the B.Sc. from the University of Cantabria, Spain, in 2020. She is currently a student of the MSc in Telecommunication Engineering in the same University. In 2019, Cristina did an internship in IT department of Solvay Chemicals Inc., where she collaborated in the development of industrial applications and services. Her research interests lie in the areas of 5G architectures and radio access networks.

Ramón Agüero, SM 2015 received his MSc in Telecommunications Engineering (1st class honors) from the University of Cantabria in 2001 and the PhD (Hons) in 2008. He is currently an Associate Professor at the Communications Engineering Department in that university. His research focuses on future network architectures, especially regarding the (wireless) access part of the network and its management. He is also interested on multihop (mesh) networks, and Network Coding. He has published more than 200 scientific papers in such areas and he is a regular TPC member and reviewer on various related conferences and journals. Ramon Agüero serves in the Editorial Board of IEEE Communication Letters (Senior Editor since 2019), IEEE Open Access Journal of the Communications Society, Wireless Networks (Springer), Mobile Information Systems (Hindawi). Dr. Agüero has supervised 5 PhDs and more than 70 BSc and MSc thesis. He is the main instructor in courses dealing with Networks, and Traffic Modeling, both at BSc and MSc levels. Since 2016, he is the Head of the IT Area (deputy CIO) at the University of Cantabria.