B. Han, V. Sciancalepore, D. Feng, X. Costa-Perez and H. D. Schotten, "A Utility-Driven Multi-Queue Admission Control Solution for Network Slicing," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, Paris, France, 2019, pp. 55-63.

doi: 10.1109/INFOCOM.2019.8737517

A Utility-Driven Multi-Queue Admission Control Solution for Network Slicing

Bin Han*, Vincenzo Sciancalepore†, Di Feng‡, Xavier Costa-Perez† and Hans D. Schotten*§

*Technische Universität Kaiserslautern, Germany
†NEC Laboratories Europe, Germany
‡Universitat Autònoma de Barcelona, Spain
§DFKI GmbH, Germany

***Abstract*—The combination of recent emerging technologies such as network function virtualization (NFV) and network programmability (SDN) gave birth to the Network Slicing revolution. 5G networks consist of multi-tenant infrastructures capable of offering leased network "slices" to new customers (e.g., vertical industries) enabling a new telecom business model: Slice-as-a-Service (SlaaS). In this paper, we aim *i*) to study the slicing admission control problem by means of a multi-queuing system for heterogeneous tenant requests, *ii*) to derive its statistical behavior model, and *iii*) to provide a utility-based admission control optimization. Our results analyze the capability of the proposed SlaaS system to be approximately Markovian and evaluate its performance as compared to legacy solutions.**

***Index Terms*—5G, network slicing, NFV, cloud service, resource management, queuing theory**

I. Introduction

Network Slicing [1] is an emerging 5G technology that allows infrastructure providers to offer “slices” of resources (computational, storage and networking) to network tenants. In this way a new business game [2] is introduced as infrastructure providers (sellers) strategically decide which tenants (buyers) get granted slices to deliver their services. Intuitively, this involves a number of challenges that fall in the economic research field, which, in turn, requires a detailed understanding of the context. In particular, the infrastructure provider may rely on this emerging technology as a means to increase its revenue sources. However, to achieve the overall revenue maximization, advanced admission control policies are required as tenants compete for a limited bunch of available resources.

In this competing environment, a brokering solution may act as a mediator between seller and buyers while providing service level agreements (SLAs) guarantees to granted running slices [3]. Admission control policies will guide the broker in the process of deciding the set of network slices that can be installed on the system and the ones to be rejected. As the number of network slices grows—as envisioned for the next few years [4]—it will be necessary to design an automated solution that dynamically decides on the received slice requests while guaranteeing a certain degree of fairness among network tenants. Indeed, network slice requests may be queued while waiting for the next available resources, or may be re-issued.

To properly design such a *slicing brokering process*, a deep understanding of the slice queuing behavior is needed that accounts, for e.g. the average slice duration (based on the slice type), the frequency of slice requests (based on the tenant), etc. This enables a Slice-as-a-Service (SlaaS) [5] solution that fully supports on-demand slices requests: tenants issue slice requests for given periods of time and decide whether to reissue the same request upon rejection based on service level agreements. Advanced slicing admission control solutions may have different policies for tenants frequently asking for short-term slices—such as Internet-of-Things (IoT), or crowded event-based network slices—as they will automatically reissue the same request in the near future, with respect to those that require only few longer network slices—such as Mobile Virtual Network Operators (MVNOs) or Industrial Network Slices [6]—which may be probably lost if not accepted. Moreover, similar as widely recognized in all kinds of queuing systems for service scheduling, tenants may be *impatient* and choose to leave for another available infrastructure provider instead of waiting in queue, especially when the expected waiting time is long. Such behavior shall also be taken into account while designing a slicing admission control solution to mitigate potential revenues loss in case of resource congestion.

While conventional admission control problems have been extensively studied in the literature, we pioneer a new stochastic model for network slicing that leverages on the multi-queuing system to optimally design an admission control of on-demand network slices as well as to orchestrate them once are accepted. This also allows to account for impatient tenant behaviors and heterogeneous network slice characteristics while, at the same time, enforcing given performance metrics, such as fairness between different tenants or between network slice types or utility-based maximization.

II. Model design

We cast our problem into a typical network slicing scenario, where the Mobile Network Operator (MNO) decides to lease infrastructure resources to tenants, willing to pay to take over the control of an independent network slice so as to deliver an end-service to their own users. Hereafter, we deeply describe our assumptions and mathematically formulate the problem.

A. Resource pool and slice types

Let us consider a single MNO that possesses a static resource pool of M different resources and offers $N = |\mathcal{N}|$ predefined types of slices. Depending on the slice type $n \in \mathcal{N}$, it costs a certain resource bundle to create and maintain a slice. Let $\mathbf{r} = [r_1, r_2, \ldots, r_M]^{\mathrm{T}}$, $\mathbf{s} = [s_1, s_2, \ldots, s_N]^{\mathrm{T}}$ and $\mathbf{c}_n = [c_{1,n}, c_{2,n}, \ldots, c_{M,n}]^{\mathrm{T}}$ denote the resource pool, the set of slices under maintenance and the resource bundle required to maintain a slice of type $n \in \mathcal{N}$, respectively. The assigned resources can be then represented as

$$\mathbf{a} \overset{\Delta}{=} [a_1, a_2, \ldots, a_M]^{\mathrm{T}} = \mathbf{C} \times \mathbf{s}, \tag{1}$$

where $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_N]$. At any time instance, the MNO cannot simultaneously maintain more slices than its resource pool may support. This constraint is expressed using the *space of resource feasibility* [7]:

$$\mathbb{S} = \{\mathbf{s} | r_m - a_m \geq 0, \quad \forall 1 \leq m \leq M\}. \tag{2}$$

Note that $\mathbb{S}$ is a finite discrete set, thus the MNO can be characterized as a finite state machine where each slice set under maintenance represents the system state $\mathbf{s} \in \mathbb{S}$.

B. Slice admission in SlaaS

We consider a certain number of tenants randomly generating network slice requests. Slices requested by a certain tenant are of the same type. For each tenant, the inter-arrival time between two requests is drawn from an exponential distribution. The request arrivals of different tenants are independent and identically distributed (i.i.d.).

Once a request for slice creation is triggered, the MNO makes a binary decision, i.e., the MNO either accepts or declines it. Upon acceptance, the requested slice is created, and continuously maintained so that a corresponding bundle of network resources is occupied until the slice is terminated (at the end of its lifetime) and the resource bundle is released. It should be noted that the constraint of space of resource feasibility forbids the MNO to accept any request when its current state is close to the border of $\mathbb{S}$. In other words, if the current MNO resource pool is close to be saturated by active slices, it does not accept additional network slice requests that might experience a service disruption. This introduces the well-known concept of *admissibility region*[1] described as

$$\mathbb{A} = \{\mathbf{s} | \mathbf{s} \in \mathbb{S}, \exists n : \mathbf{s} + \Delta\mathbf{s}_n \in \mathbb{S}\}, \tag{3}$$

where $\Delta\mathbf{s}_n$ is the *unit slice incremental vector* of type n

$$\Delta\mathbf{s}_n = [\underbrace{0, \ldots, 0}_{n-1}, 1, \underbrace{0, \ldots, 0}_{N-n}], \quad n \in \{1, 2, \ldots, N\}. \tag{4}$$

We assume that the lifetime of every slice is an i.i.d. exponentially distributed variable and the expected lifetime depends on the slice type. We also consider that the MNO makes every decision according to a consistent slicing policy, i.e., the decision depends only on the type of requested slice n and the current system state $\mathbf{s}$ that defines the current set of slices under maintenance.

[1]The admissibility region has been exhaustively studied in the literature for different use cases and scenarios. We refer the reader to [8], where a stochastic admissibility region is derived for a network slicing admission control.

C. Delayed reattempt upon request denial

If a request for slice creation is declined—because of a temporary shortage of available resources due to many other active slices—the tenant is not able to obtain the requested slice immediately. Instead, its request may be sent to the MNO again for a reconsideration after some delay with the hope that some running slice has expired (i.e., resources have been released). Generally, there are two critical features of the delaying mechanism, which should be taken into account: *i*) resource efficiency and *ii*) fairness. The former requires that the chosen mechanism purses the resource pool utilization maximization whereas the latter requires that the expected delay for different requests is normalized.

Two categories of approaches are commonly used to solve this kind of problem:

Random delay. Every declined request is re-proposed to the MNO after a random delay. This approach provides a good fairness, but generates extra signaling overhead in the control plane being not able to provide the discipline of "First Come, First Served" (FCFS), as described in the next section.

Queuing. Declined requests wait in one or multiple queue(s) for the next opportunity during the MNO's decisional process. This is the most common solution in cloud service scheduling.

Hereafter, we show how a multi-queuing system may be fully exploited to provide insights on the system behaviors and pave the road towards a slicing orchestration solution.

III. Network slicing queuing

In the literature a number of various disciplines have been studied to serve the request queues. Among the others, the most common policies are *i*) First come, first served (FCFS), *ii*) Last come, first served (LCFS), *iii*) Random selection for service (RSS) and *iv*) Priority-based (PR). All of them analyze different behaviors and are used to achieve distinct performance metrics. For instance, the LCFS is used to reduce the fairness whereas the priority-based is implemented when there is some high-level preference of the MNO to be considered. RSS shows huge complexity in the implementation without bringing any significant advantage with respect to the others. Hereafter, we focus on the FCFS case. However, any other discipline may be easily adapted to our analysis.

A. Queuing schemes

We differentiate the queuing systems into two different categories: *i*) single-queue and *ii*) multi-queue systems. When considering the single-queue, only one queue is implemented for all declined requests that need to wait for the next acceptance opportunity. Conversely, the multi-queue system implements multiple queue for declined requests. Specifically, such queues may show different features. We consider homogeneous-mixed queues, wherein each queue consists of requests for slices of different types, and heterogeneous queues, where each queue is specified for only one unique slice type. We next show a simple case-study to justify that the queuing system is suitable for this kind of problems.

B. Resource efficiency: a simple case-study

Consider a simplified case where $M = 1$, $N = 2$, $\mathbf{r} = [1]$, $\mathbf{c}_1 = [0.6]$, $\mathbf{c}_2 = [0.2]$ and $\mathbf{s} = [1, 0]^T$. The first four requests awaiting in the queue(s) are in the sequential order $[1, 1, 2, 2]$. The MNO is taking a greedy strategy that intends to accept all requests received so far the resource pool supports.

Both in the schemes with a single queue and two homogeneous queues, the MNO fails to accept requests of type 2 as the type 1 requests are preventing their acceptance. Hence, it has to wait until the currently active slice of type 1 is released before it can accept the next request in the queue, although it has both enough idle resource and the intention. The heterogeneous multi-queue scheme, in contrast, enables the MNO to fully utilize its resource pool as shown in Fig. 1.

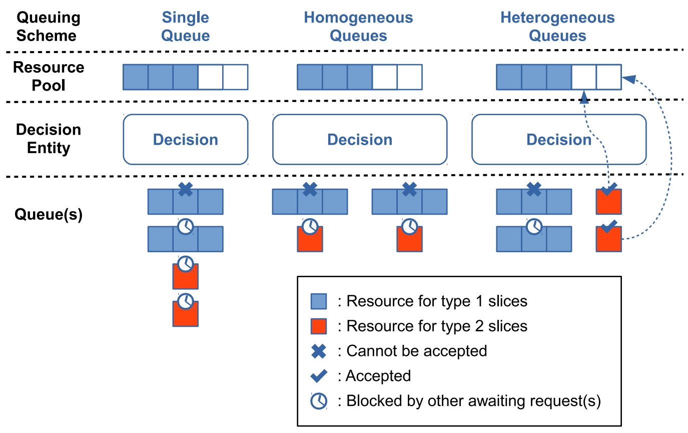


Fig. 1: A simple case study on different queuing schemes.

Obviously, both the single-queue and the homogeneous multi-queue schemes can also overcome this issue by introducing a "queue-jumping" mechanism. However, this may require an extra design of (more complex) logic that automatically (and dynamically) decides which request is allowed to jump in the queue(s). Therefore, in this study we consider the scheme with N FCFS heterogeneous queues.

IV. Heterogeneous multi-queue admission control

Based on the heterogeneous multi-queue scheme, we propose in this section a novel code to present the MNO's preference for different slice types in variable states, a multi-queue admission controller for SlaaS, and analyze its queue model.

A. Slice-type preference encoder

Differing from existing studies that do not consider queuing and the single-queue scheme, in the multi-queue scheme, the MNO may receive multiple requests for slices of different types simultaneously. Therefore, instead of making a simple binary decision of accepting or declining one request, it has to either choose one from the simultaneously arriving requests to accept while declining the rest ones, or decline all of them. Especially, with heterogeneous queues, the MNO's preference for some request queue(s) over the others implies its proclivity to some slice type(s) against the others.

For an MNO that offers N different slice types to tenants for request, we can encode an arbitrary preference of the MNO into a *preference vector* of length $N + 1$:

$$\Phi = [\varphi_1, \varphi_2, \dots, \varphi_{N+1}], \tag{5}$$

which is a permutation of $\{0, 1, 2, \dots, N\}$. The earlier a queue number $1 \le n \le N$ occurs in Φ, the more likely the MNO prefers slice type n over the others. Note that $n = 0$ denotes reserving resource for potential opportunities in future, so that all requests in the queues with values occurring in Φ after 0 will not be served by the MNO at all.

While being in states on (or close to) the border of space of resource feasibility $\mathbf{s} \in \mathbb{S} - \mathbb{A}$, the MNO cannot accept further request from any queue, hence the preference does not make any impact. Thus, we focus on the admissibility region $\mathbb{A}$ and assume that the MNO's preference is consistent and depends only on its current state $\mathbf{s} \in \mathbb{A}$. Thus, we can characterize the MNO's admission strategy with a $(N + 1) \times |\mathbb{A}|$ *preference matrix* as the following

$$\begin{aligned}\boldsymbol{\Phi} &= [\Phi_1, \Phi_2, \dots, \Phi_{|\mathbb{A}|}] \\ &= \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \cdots & \phi_{1,|\mathbb{A}|} \\ \phi_{2,1} & \phi_{2,2} & \cdots & \phi_{2,|\mathbb{A}|} \\ \vdots & \vdots & \ddots & \vdots \\ \phi_{N+1,1} & \phi_{N+1,2} & \cdots & \phi_{N+1,|\mathbb{A}|} \end{bmatrix},\end{aligned} \tag{6}$$

where each column Φ_i represents the MNO's preference for different slice types in a specific feasible state in $\mathbb{A}$.

B. Mechanism overview

Let l_n denote the length of the n^{th} queue, the decision entity executes the algorithm described in Fig. 2. The MNO keeps waiting for incoming tenant issues and responses to them upon issue arrivals. If the tenant issues to release a slice of its own, the MNO always releases it. If the tenant requests for a new slice, the request will be pushed into the corresponding queue with respect to the type of requested slice. After responding to the issue, the MNO will recursively serve the request queues in a sequence determined by its admission strategy and active slice set, until no more waiting request can be accepted. Then it stops serving the queues and waits for the next tenant issue.

V. Network slicing controller design

We analyze different characteristics of the conventional queuing models, highlighting the novel features applied to our model while designing the network slicing controller. This helps to shed the light on the main advantages and limitations of our novel admission control model.

A. Analysis of inter-acceptance time

We consider request arrivals of every slice type as an independent Poisson process, so that the inter-arrival time between requests in every queue is an independent exponential random process. Conversely, the request acceptance rate of

```
Initialize with certain $N$, $\mathbb{S}$, $\mathbb{A}$, $\boldsymbol{\Phi}$ and $\mathbf{s}$;
while True do                                                  Main loop
    Wait for the next incoming tenant issue;
    if Slice of type n released then                          Releasing a slice
        $\mathbf{s} \leftarrow \mathbf{s} - \Delta\mathbf{s}_n$;
    else if Slice of type n requested then                     Request arrives
        $l_n \leftarrow l_n + 1$;
    end
    while $\mathbf{s} \in \mathbb{A}$ do          Recursively serving the queues until blocked
        $\tilde{\mathbf{s}} \leftarrow \mathbf{s}$;
        Find the current preference vector $\Phi$ according to $\boldsymbol{\Phi}$ and $\mathbf{s}$;
        for $1 \le n \le N$ do                                 Serve queues w.r.t. preference
            if $\varphi_n = 0$ then                            Omitting queues after 0
                break;
            else if $l_n > 0$ AND $(\mathbf{s} + \Delta\mathbf{s}_n) \in \mathbb{S}$ then    Acceptance
                $l_n \leftarrow l_n - 1$;
                $\mathbf{s} \leftarrow \mathbf{s} + \Delta\mathbf{s}_n$;
            end
        end
        if $\tilde{\mathbf{s}} = \mathbf{s}$ then                      Blockage detection
            Break;
        end
    end
end
```

Fig. 2: The multi-queue slice admission controlling algorithm.

every queue is jointly determined by the slice releases of all types, and the MNO's preference strategy.

Theorem 1. *Consider a heterogeneous multi-queue slice admission controller that executes the algorithm in Fig. 2 with a consistent preference matrix. The acceptance in different queues are mutually independent Poisson processes, if: 1) the arrivals of new requests and releases of active slices are mutually independent Poisson processes for every individual slice type; 2) the arrivals of different slice types are mutually independent from each other, the releases of different slice types are mutually independent from each other.*

Proof. First, extend the system (MNO) state $\mathbf{s}$ with all queue lengths to obtain the *controller state* $\hat{\mathbf{s}} = [\mathbf{s}, l_1, l_2, \dots, l_N]$, and therefore the infinite discrete domain $\hat{\mathbb{A}} = \mathbb{A} \times \mathbb{N}^N$. Let the bijection $\mathbb{A} \leftrightarrow \{1, 2, \dots, |\mathbb{A}|\}$ denoted by $I = I_{\mathbb{A}}(\mathbf{s})$, we call $\hat{\mathbf{s}} \in \hat{\mathbb{A}}$ a *transient* state if $\exists n \in \mathcal{N}$ such that:

$$\begin{cases} \phi_{I,k} \neq 0, \quad \forall k < n; & (7) \\ l_n > 0; & (8) \\ \left(\mathbf{s} + \Delta\mathbf{s}_{\phi_{n,I}}\right) \in \mathbb{A}. & (9) \end{cases}$$

Otherwise, we call $\hat{\mathbf{s}}$ a *steady* state. According to the algorithm in Fig. 2, when the controller is in a transient state it always accepts a request in its queues immediately and therefore keeps jumping to another state until it reaches a steady state. Every transient state leads to one and only one certain steady state. On the other hand, the controller can reasonably (but not always) leave a steady state only when a new request arrives or a slice is released.

Thus, given a certain sequence of request arriving and slice releasing events in the next period, we can obtain the transition path of the controller state, and therewith determine whether the first awaiting request in an arbitrary queue will be accepted during that period. Denote the time that the first awaiting request in the nth queue still has to wait until it is accepted as $t_{\mathrm{w},n}$, it yields that

$$\mathrm{Prob}(t_{\mathrm{w},n} > T) = 1 - \prod_{\mathbf{e} \in \mathbb{E}_n} \mathrm{Prob}(Arr(T) = \mathbf{e}), \tag{10}$$

where $\mathbb{E}_n$ is the set of all event sequences that can lead to an acceptance of request in the nth queue, and $Arr(T)$ denotes the event sequence arriving in the next period of T. As the request arrivals and releases of different slice types are mutually independent Poisson processes, we know that all $\mathbf{e} \in \mathbb{E}_T$ are also approximately Poissonian (proven as a feature of *dependent trials* [9], [10]). Thus, due to the Markovian behavior of Poisson processes, we can write the following

$$\begin{aligned} \mathrm{Prob}(Arr(T) = \mathbf{e}) &= \mathrm{Prob}(Arr(T + t) = \mathbf{e} \mid Arr(t) \neq \mathbf{e}) \\ &\qquad \forall [\mathbf{e}, t, T] \in \left(\mathbb{E}_T \times \mathbb{N}^2\right), \end{aligned} \tag{11}$$

and thus

$$\begin{aligned} &\mathrm{Prob}(t_{\mathrm{w},n} > T + t) \\ =& 1 - \prod_{\mathbf{e} \in \mathbb{E}_n} \mathrm{Prob}(Arr(T + t) = \mathbf{e} \mid Arr(t) \neq \mathbf{e}) \\ =& \mathrm{Prob}(t_{\mathrm{w},n} > T + t \mid t_{\mathrm{w},n} > t), \quad \forall [t, T] \in \mathbb{N}^2. \end{aligned} \tag{12}$$

Eq. (12) implies that the remaining waiting time for acceptance of the first request in queue n is *memoryless*. Due to the fact that the only two classes of memoryless distributions are exponential (continuous) and geometric (discrete) distributions, we can assert that the request acceptance in every queue is a Poisson process. □

B. Queuing-theoretic analysis

While considering both request arrivals and request acceptances (service) as Poisson processes, every request queue is a classic M/M/1 queuing system, known as single-server birth-death system [11]. Hence, many features of birth-death model can be directly applied.

1) Little's Formula: For slice type (queue) n, given its request arrival rate λ_n, according to the famous Little's formula [12] there is

$$L_n = \lambda_n \overline{W}_n, \tag{13}$$

where L_n and $\overline{W}_n$ represent the mean length of queue n and the average waiting time in queue n, respectively.

2) Steady Queue State Probability: Given the request arrival rate λ_n and acceptance rate μ_n of queue n, the probability that the queue steadily consists of l requests at an arbitrary time instant is geometrically distributed, i.e.,

$$p_n(l) = (1 - \rho)\rho^l, \tag{14}$$

where $\rho_n = \lambda_n / \mu_n < 1$ is the *work load rate* of queue n.

3) Waiting Time Distribution: The probability density function (PDF) of an arbitrary type-n request's waiting time is

$$f(W_n) = \begin{cases} 0 & W_n < 0 \\ (\mu_n - \lambda_n)e^{-(\mu_n - \lambda_n)W_n} & W_n \geq 0 \end{cases}, \tag{15}$$

and the cumulative density function (CDF) is

$$F(W_n) = \begin{cases} 0 & W_n < 0 \\ 1 - e^{-(\mu_n - \lambda_n)W_n} & W_n \geq 0 \end{cases}. \tag{16}$$

C. Extension: impatient tenants

From Eqs. (13–16) it is clear that both L_n and W_n converge only when $\lambda_n < \mu_n$. Otherwise, when the request acceptance rate is lower than the arrival rate in queue n, the queue length will infinitely increase, and therefore also the mean waiting time. This is known as the necessary and sufficient condition of statistical equilibrium in queuing processes, as stated and proven by *Kendall* in work [13].

However, in a real slice admission controller, there are various situations where $\lambda_n \geq \mu_n$ for some n, including cases

- when the controller is specified with an inappropriate strategy, so that requests in the queue n is rarely or even never accepted despite of resource feasibility;
- when the release rates of active slices are low, so that the resource pool fails to support a sufficiently high μ_n regardless of any admission strategy.

There are two mechanisms that prevent queuing systems from such divergence. On the one hand, the system may force to truncate a queue at some maximal length, and forbid this queue to take any new request before it is shortened. On the other hand, the clients may lose patience while waiting, and leave the queues before being served (e.g., for looking for some other MNO with resource availability). In the scenario of SlaaS, the system (MNO) is probably very cautious with refusing requests, while the waiting time can be critical to the customers (tenants). Therefore, here we consider no queue truncation but queues with impatience.

Usually, impatience in queues can occur in three different behaviors: *i)* balking, i.e. customers being reluctant to join a queue upon arrival, *ii)* reneging, i.e. customers leaving the queue after joining and waiting, and *iii)* jockeying from long lines to shorter ones. As the heterogeneous multi-queue design disables jockeying, here we consider the balking and reneging phenomena.

Balking Model. The phenomenon of balking can be modeled in such a way, that every arrival request of slice type n enters the queue with a probability b_n, which is a monotonically decreasing function of the current queue length l_n. *Ancker* and *Gafarian* have proposed two different balking models in [14], [15]. The first model considers a linear balking factor $1 - b_n = l_n/l_{n,\max}$, where $l_{n,\max}$ is the upper bound of l_n for queue truncation. The second one considers a non-linear balking factor as follows

$$1 - b_n = \begin{cases} 0 & l_n = 0 \\ 1 - \beta_n/l_n & l_n \in \mathbb{N}^+ \end{cases}, \tag{17}$$

where $\beta_n \in [0, 1]$ measures the willingness of tenants requesting type-n slices to wait. In cases that the tenant has knowledge about μ_n, *Shortle* et al. suggest another non-linear balking model $1 - b_n = 1 - e^{-\beta_n l_n/\mu_n}$ where $\beta_n > 0$ [11]. Here we consider the hyperbolic balking model described by Eq. (17).

Reneging Model. The phenomenon of reneging can be modeled by randomly assigning an individual maximal waiting time to every request when it joins the queue. The request will leave the queue after that maximal waiting time if it has not been accepted yet. Following *Ancker* and *Gafarian* [15], we consider the maximal waiting time for every type-n request as an exponential random variable $W_{\max,n} \sim \mathrm{Exp}(\alpha_n)$, where $1/\alpha_n > 0$ is the mean maximal waiting time in queue n.

D. Performances with balking and reneging

It should be noted that the balking and reneging processes are with memory, leading to a non-Markovian behavior of request acceptances. However, under low balking and reneging rates, this impact can be negligible and the acceptance process can still be approximated as Poissonian. When the balking and reneging rates rise to significant levels, the memory of acceptance process shall be considered, as demonstrated in Section VII-A by means of simulations.

Under a combination of hyperbolic balking and exponential reneging, the steady state probability of having l requests in the queue n is

$$p_n(l) = \begin{cases} \frac{1}{1+(\delta_n)^{1-\gamma_n/2}[\Gamma(\gamma_n)/\beta_n]I_{\gamma_n}(2\sqrt{\delta_n})} & l = 0 \\ \\ \frac{\delta_n^l p_n(0)}{\beta_n (l-1)! \prod_{j=0}^{l-1}(\gamma_n + j)} & l \in \mathbb{N}^+ \end{cases}, \tag{18}$$

where $\gamma_n = \mu_n/\alpha_n$, $\delta_n = \lambda_n\beta_n/\alpha_n$, $I_{\gamma_n}(\cdot)$ is the modified Bessel's function of the first kind and order γ_n.

Meanwhile, we are interested in three different distributions of waiting time spent in a queue n: *i)* $f_\mathrm{a}(W_n)$ for requests that are eventually accepted, *ii)* $f_\mathrm{r}(W_n)$ for requests that renege and *iii)* $f_\mathrm{q}(W_n)$ for all requests that join the queue. Let us define A_n and J_n as the events of request being accepted and joining the queue n, respectively. There are

$$P(A_n) = \frac{[1 - p_n(0)]\beta_n\gamma_n}{\delta_n}, \tag{19}$$

$$P(A_n, J_n) = \frac{[1 - p_n(0)]\beta_n\gamma_n}{\delta_n} - p_n(0), \tag{20}$$

$$P(A_n|J_n) = \frac{\Gamma(\gamma_n + 1)I_{\gamma_n}\left(2\sqrt{\delta_n}\right) - \left(\sqrt{\delta_n}\right)^{\gamma_n}}{\sqrt{\delta_n}\Gamma(\gamma_n)I_{\gamma_n - 1}\left(2\sqrt{\delta_n}\right) - \left(\sqrt{\delta_n}\right)^{\gamma_n}}. \tag{21}$$

It can be obtained that

$$f_\mathrm{a}(W_n) = \frac{p_n(0)\lambda_n\beta_n e^{-(\mu_n+\alpha_n)W_n} I_1\left[2\sqrt{\delta_n(1 - e^{-\alpha_n W_n})}\right]}{P(A_n, J_n)\sqrt{\delta_n\left(1 - e^{-\alpha_n W_n}\right)}}, \tag{22}$$

$$f_\mathrm{r}(W_n) = \alpha_n e^{-\alpha_n W_n} \frac{1 - P(A_n|J_n)g(W_n)}{1 - P(A_n|J_n))}, \tag{23}$$

$$f_\mathrm{q}(W_n) = P(A_n|J_n)\left[f_\mathrm{a}(W_n) - \alpha_n e^{-\alpha_n W_n} g(W_n)\right] + \alpha_n e^{-\alpha_n W_n}, \tag{24}$$

where $g(W_n) = \int_0^{W_n} e^{\alpha_n \xi} f_\mathrm{a}(\xi)\mathrm{d}\xi$.

The expectations of waiting times are therefore

$$\overline{W}_{\mathrm{a},n}\frac{p_n(0)}{P(A_n,W_n)}\sum_{i=1}^{+\infty}\left[\frac{\delta_n^i}{i!}\prod_{j=1}^{n}(\gamma_n+j)\right]\sum_{k=1}^{i}\frac{1}{\gamma_n+k},\tag{25}$$

$$\overline{W}_{\mathrm{r},n}=\frac{1}{\alpha_n}-\frac{P(A_n|W_n)\overline{W}_{\mathrm{q},n}}{1-P(A_n|W_n)},\tag{26}$$

$$\overline{W}_{\mathrm{q},n}=\frac{1-P(A_n|W_n)}{\alpha_n}.\tag{27}$$

VI. Strategy optimization

In slice admission control, there are various performance metrics that may include: the overall network utility rate, the admission rate and the average request waiting time.

The network utility of a slice can be differently defined, such as the periodical payment that the MNO receives from the tenant, or the generated network throughput, etc. It is common to consider the utility rate of a slice as determined by the slice type, and the overall network utility rate at any time instant t as the sum of utility rates of all slices under maintenance:

$$u_\Sigma(t)=\sum_{n=1}^{N}s_n(t)u_n,\tag{28}$$

where $s_n(t)$ is the number of type-n slices under maintenance at time t, and u_n is the utility rate of every type-n slice. In long term, the average overall network utility rate can be estimated from the acceptance and releasing rates of different slice types:

$$\overline{u}_\Sigma=\sum_{n=1}^{N}\frac{\mu_n u_n}{\eta_n},\tag{29}$$

where η_n is the releasing rate per type-n slice.

The average waiting time of all requests in queues is

$$\overline{W}_{\mathrm{q}}=\frac{\sum\limits_{n=1}^{N}\overline{W}_{\mathrm{q},n}L_n}{\sum\limits_{n=1}^{N}L_n}.\tag{30}$$

The overall admission rate is the following

$$\overline{P}(A)=\frac{\sum_{n=1}^{N}\lambda_nP(A_n)}{\sum_{n=1}^{N}\lambda_n}.\tag{31}$$

All three criteria are determined by the request behavior parameters $\alpha_n,\beta_n,\lambda_n$ and the acceptance rate μ_n. Given a certain combination of $[\alpha_n,\beta_n,\lambda_n,\eta_n]$, where $1/\eta_n$ is the average lifetime of type n slices, μ_n is uniquely determined by the MNO's strategy, i.e. by the preference matrix $\mathbf{\Phi}$. Hence, with consistent behaviors of request arrival and slice releasing, we can optimize either of them by selecting the best $\mathbf{\Phi}$.

A major challenge for analysis exists in the complex relation between the acceptance rates $[\mu_1,\mu_2,\dots,\mu_N]$ and the strategy $\mathbf{\Phi}$, as $\mathbf{\Phi}$ does not directly imply the MNO's action or statistics, but only its preference.

Nevertheless, if the steady-state probability of queue lengths $p_n(l)$, as defined in Eq. (14), is known or measurable for all $n\in\mathcal{N}$, we can estimate μ_n for all n with respect to $\mathbf{\Phi}$ and the initial state $\mathbf{s}_{\mathrm{init}}$ as follows.

First, define a bijection $\mathbb{S}\leftrightarrow\{1,2,\dots,|\mathbb{S}|\}$ as $J=J_{\mathbb{S}}(\mathbf{s})$ where $J_{\mathbb{S}}(\mathbf{s})=I_{\mathbb{A}}(\mathbf{s})$ for all $\mathbf{s}\in\mathbb{A}$. Then extend the definitions in Eqs. (4), (6) and (14) with

$$\Delta\mathbf{s}_0=\underbrace{[0,0,\dots,0]}_{N},\tag{32}$$

$$\tilde{\phi}_{i,j}=\begin{cases}0 & j>|\mathbb{A}|\\ \phi_{i,j} & j\le|\mathbb{A}|\end{cases},\forall i\in\{1,2,\dots,N+1\},\tag{33}$$

$$p_0(0)=0,\tag{34}$$

respectively. The probability of state transition from any $\mathbf{s}\in\mathbb{S}$ to $\mathbf{s}+\Delta\mathbf{s}$ can be then calculated as

$$\mathrm{Prob}(\mathbf{s}\rightarrow\mathbf{s}+\Delta\mathbf{s}_n)=\prod_{k=1}^{n-1}p_{\tilde{\phi}_{k,J}}(0)(1-p_{\tilde{\phi}_{n,J}}(0)).\tag{35}$$

Thus, when the initial state $\mathbf{s}_{\mathrm{init}}$ is known, we can obtain the long-term probability distribution of system state $\mathbf{s}$ as

$$\mathrm{Prob}(\mathbf{s}_j\mid\mathbf{s}_{\mathrm{init}}=\mathbf{s}_i)=\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K}[\mathbf{\Psi}^k]_{i,j},\tag{36}$$

where $\mathbf{\Psi}$ is the transition matrix:

$$\mathbf{\Psi}=\begin{bmatrix}\Psi_{1,1} & \Psi_{1,2} & \dots & \Psi_{1,|\mathbb{S}|}\\ \Psi_{2,1} & \Psi_{2,2} & \dots & \Psi_{2,|\mathbb{S}|}\\ \vdots & \vdots & \ddots & \vdots\\ \Psi_{|\mathbb{S}|,1} & \Psi_{|\mathbb{S}|,2} & \dots & \Psi_{|\mathbb{S}|,|\mathbb{S}|}\end{bmatrix},\tag{37}$$

and $\Psi_{i,j}=\mathrm{Prob}(\mathbf{s}_i\rightarrow\mathbf{s}_j)$.

More generally, if not the exact value but the probability distribution of the initial state is available as $P_{\mathrm{init}}=[p_{\mathrm{init}}(\mathbf{s}_1),p_{\mathrm{init}}(\mathbf{s}_2),\dots,p_{\mathrm{init}}(\mathbf{s}_{|\mathbb{S}|})]$, the long-term probability distribution $\mathbf{s}$ is the following

$$\mathrm{Prob}(\mathbf{s}_j\mid P_{\mathrm{init}})=\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K}\sum_{i=1}^{|\mathbb{S}|}p_{\mathrm{init}}(\mathbf{s}_j)[\mathbf{\Psi}^k]_{i,j}.\tag{38}$$

We can obtain the expected active slice number $\overline{s}_n$ of every slice type n as a function of $\mathbf{\Psi}$ and thus, as a function of $\mathbf{\Phi}$. Now, recalling Eqs. (28–29) it yields that

$$\overline{s}_n=\frac{\mu_n}{\eta_n},\tag{39}$$

and then we can write the following

$$\begin{aligned}\mu_n=\frac{\overline{s}_n}{\eta_n}&=\frac{\sum\limits_{\mathbf{s}\in\mathbb{S}}\mathrm{Prob}(\mathbf{s}\mid P_{\mathrm{init}})s_n}{\eta_n}\\ &=\frac{1}{\eta_n}\sum_{\mathbf{s}\in\mathbb{S}}\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K}\sum_{i=1}^{|\mathbb{S}|}p_{\mathrm{init}}(\mathbf{s}_j)[\mathbf{\Psi}^k]_{i,j}.\end{aligned}\tag{40}$$

Based on this analytical expression, we are able to optimize $[\mu_1,\mu_2,\dots,\mu_n]$ with respect to $\mathbf{\Phi}$. However, it is evident that Eq. (40) is non-convex w.r.t. $\mathbf{\Phi}$, which prohibits analytical solution of the global optimum. On the other hand, the overall domain size of $\mathbf{\Phi}$ is $2^{(N+1)|\mathbb{A}|}$, which can assume unaffordable high values for any realistic dimension of $|\mathbb{A}|$ in practical networks, making the exhaustive search impossible. This is an integer linear programming (ILP) problem that is proven to be NP-Hard, therefore advanced machine learning and heuristic search methods are needed to solve it with affordable efforts of computation.

VII. Numerical simulations

To carry out simulations in a consistently specified environment, we consider an MNO with a two-dimensional ($M = 2$) normalized resource pool $\mathbf{r} = [r_1, r_2] = [1, 1]$. $N = 2$ slice types are defined in two service demand scenarios, as shown in Tab. I. Note that α_n and β_n are only applicable when the simulation considers balking and reneging, respectively.

Type (n)	$\mathbf{c}_n$	λ_n	$1/\eta_n$	u_n	α_n	β_n
1	$[0.01, 0.05]$	2 (Scenario 1)	5	1	1	0.02
		6 (Scenario 2)				
2	$[0.2, 0.04]$	0.5 (Scenario 1)	2	10		
		1.5 (Scenario 2)				

TABLE I: Specifications of two reference slice types

A. *Verification of geometric IAT distribution*

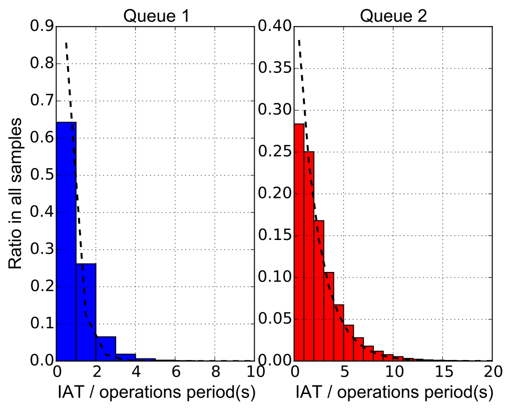
In case of patient tenants, Theorem 1 can also be verified through numerical simulations. We take the slice specifications in scenario 1, disable balking and reneging events, and randomly generate 500 slicing strategies. For each strategy, 20 rounds of Monte-Carlo tests are executed. In each testing round, an MNO with a 2-queue slice admission controller is initialized to a random but fully resource-utilized state, and then operates under the consistent strategy for 40 operations periods. Then we investigate the distribution of inter-acceptance time (IAT) for each queue, and fit the measurements with geometric distributions, which is the discrete-time version of exponential distribution. A sample result is shown in Fig. 3(a), where a good fitting performance can be observed.

To quantitatively evaluate the fitness, we compute the Kullback-Leibler divergence (KLD) [16] for every strategy:

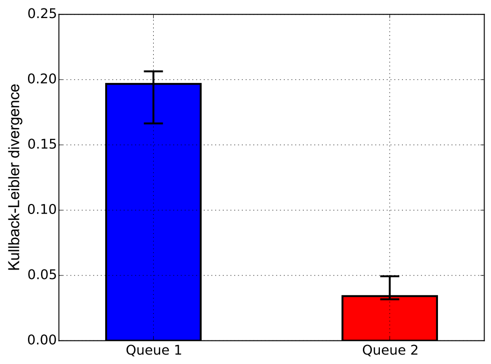
$$D_{\mathrm{KL}}(P_{\mathrm{IAT}} \mid \mathrm{Geom.}) = \sum_{k=0}^{\infty} p_{\mathrm{IAT}}(k) \log \frac{p_{\mathrm{IAT}}(k)}{(1-\hat{p})^k \hat{p}}, \qquad (41)$$

where $p_{\mathrm{IAT}}(k)$ is the empirical probability mess function (PMF) of the measured IAT, and $(1-\hat{p})^k\hat{p}$ is the geometric PMF with fitted parameter $\hat{p}$. KLD is an indicator of fitness between two distributions, which equals 0 for two identical distributions and approaches to 1 for two completely irrelevant distributions. The KLD distribution over all 500 tested random strategies is depicted in Fig. 3(b), which shows a satisfactory fitness for both queues (slice types).

Furthermore, to verify the impact of impatient tenants' behavior, we activate the mechanisms of balking and reneging, and repeat the aforementioned simulation procedure in both scenarios 1 and 2. The results are illustrated in Fig. 4. Compared to the case of patient tenants, we can observe an increase of KLD in both scenarios here, especially in scenario 2, confirming our assertion that the behaviors of balking and



(a) The distribution of inter-acceptance time in two different queues under a random strategy, fitted as geometric distribution.



(b) The Kullback-Leibler divergence of fitting the IAT distribution as geometric distribution, 500 random strategies tested.

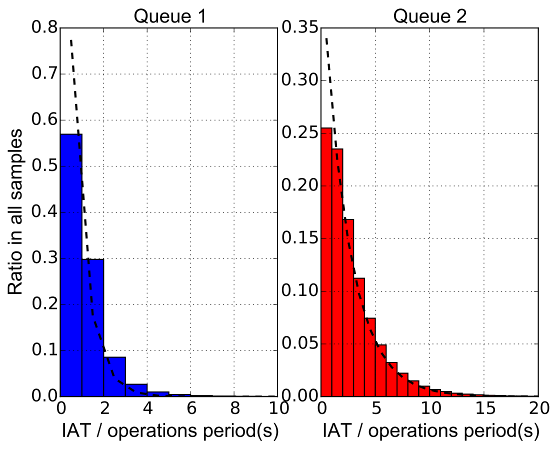
Fig. 3: The IAT of every individual queue under an arbitrary strategy is geometrically distributed.

reneging will remove the Markovian feature of the system. However, when the balking and reneging rates are low (e.g., when the queues are short such like in scenario 1), such impact can be slight enough to be neglected.

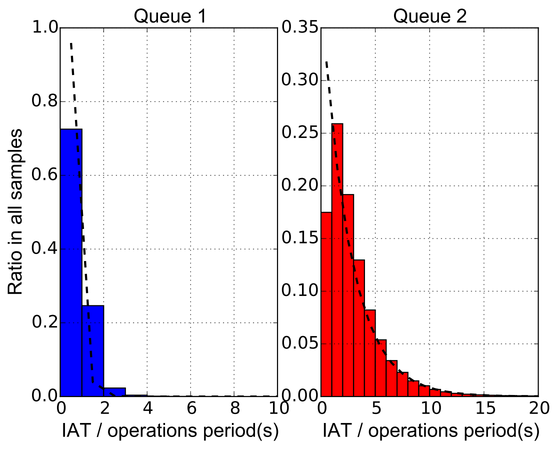
B. *Evaluation of the proposed controller*

To verify the effectiveness and potential in optimization of the proposed multi-queue slice admission controlling mechanism, we generate 10 000 random strategies, and measure all three above-mentioned performances metrics $\overline{u}_{\Sigma}$, $\overline{W}_{\mathrm{q}}$ and $\overline{P}(A)$ for every strategy in both reference scenarios 1 and 2. Similar to the last tests, every strategy is evaluated through a 20-round Monte-Carlo test where each round begins with a random initial state and lasts 40 operations periods. Impatient tenants are considered.

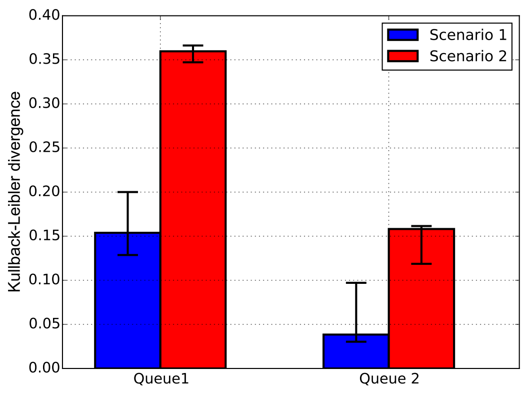
To provide benchmarks, we test the controller with two specific "naïve" strategies: *Prefer Type 1*: the preference vector is $[1, 2, 0]$ at all system states; *Prefer Type 2*: the preference vector is $[2, 1, 0]$ at all system states. Moreover, we implement and test a simple "greedy" single-queue slice admission



(a) IAT distributions in scenario 1 under a random strategy, fitted as geometric distributions.



(b) IAT distributions in Scenario 2 under a random strategy, fitted as geometric distributions.



(c) The KLD of fitting the IAT distribution as geometric distribution, in different scenarios.

Fig. 4: Balking and renaging lead to non-Poisson admissions, the impact increases with the balking and renaging rates.

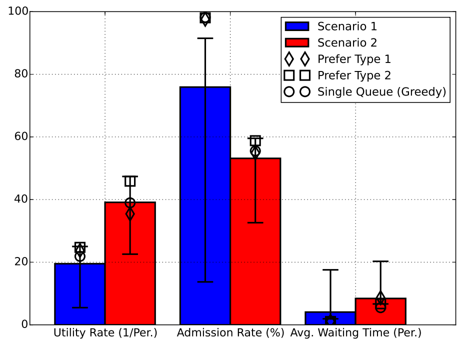


Fig. 5: Performance distribution of the proposed multi-queue slice admission controller with 10 000 random strategies, in comparison to selected benchmarks.

controller that always accepts the first request in its queue regardless of type, as long as the resource pool supports.

The results are illustrated in Fig. 5. It can be observed that the multi-queuing controller, when specified with an appropriate strategy, outperforms the greedy single-queue solution in admission rate, especially when the demand is dense and queues are congested. However, it shall be noted that the performances highly rely on the selection of strategy, leading to a critical necessity of strategy optimization.

VIII. Further discussion

In practical wireless networks, both the dynamics of resource availability (e.g. channel fading) and the resource elasticity of active slices must be taken into account. The model in this paper is an approximation with a static resource pool $\mathbf{r}$ and rigid slices, which holds in long-term with appropriate dynamic scheduling to multiplex slices. Note that such a slice multiplexing implicitly enables slice *overbooking* with a risk to break SLAs [17], [18]. The challenge of balancing the multiplexing gain and the overbooking risk in heterogeneous multi-queue admission control settings deserves future study.

It shall also be noticed that the assumptions of Poisson arrivals/releases may not hold in some practical service scenarios. In this case, the queues are not $M/M/1$ systems and cannot be considered as continuous-time Markov systems. Nevertheless, as pointed out in [11], many such continuous-time non-Markov processes can be easily transformed into discrete-time Markov chains by observing only the state transitions. Therefore, the analyses given above also apply to most scenarios with non-Poisson request arrivals/releases.

IX. Related work

We summarize in the following the main research efforts in the literature on the topic of Slice-as-a-Service, queuing theory for cloud services and network slicing admission control.

An overview on multi-tenancy service and 5G network slicing is given in [3] from perspectives of architecture and standardization, introducing the novel concept of *network slice* *broker* which executes the admission control. Different attempts have been made in [5], [8] and [19] to demonstrate how admission control can benefit the network resource utilization.

While we have considered network slicing in a generic and abstracted view, which is generally applicable in both radio access network (RAN) and core network (CN) domains, recently there has been a dense specific research interest for RAN slicing and its impact on radio resource management (RRM). On that [20] and [21] provide interesting solutions for efficient resource management and orchestration. From the perspective of slicing admission strategy optimization, the methods reported in [5], [7], [8] can be worthwhile to refer. Although all these works only consider a binary decision mechanism where declined requests simply vanish instead of being served after a delay, the algorithms deployed by them to solve ILP problems will inspire future development of model-less heuristic strategy optimizers for the proposed multi-queue slice admission controller.

SlaaS shall be considered as a specific type of public cloud environment, where service sessions can be categorized into multiple types with significantly heterogeneous resource demands. Queuing theory has been widely applied for cloud computing services to model the statistics of service demand and delivered quality of service (QoS), such as [22] and [23]. Especially, service schedulers with heterogeneous queues for different service types are discussed in [24] and [25]. These models provide valuable reference views in addition to the model proposed in this paper. Finally, balking and reneging behavior of impatient clients in queuing systems are extensively studied in [26], [27].

Differing from the aforementioned works wherein a "strategy" usually represents the decision as a function of the system state, our study proposes a novel mechanism of multi-queuing slice admission control where the slicing strategy represents the MNO's preference of slice types in different system states. Besides, out paper also considers impatient tenants, which, from the best of our knowledge, has never been investigated in SlaaS environments.

X. Conclusion

The network slicing paradigm plays a key-role in the next generation of networks design. However, it involves a number of challenges while devising an admission control solution that takes into account complex network tenants behaviors.

In this paper, we have proposed a multi-queue-based controller that automatically accounts for tenants waiting to get their requests network slices with given request frequency and patience characteristics. Our results validate the proposed model showing that unexpected tenants behaviors may be properly addressed with advanced admission control policies.

Acknowledgments

This work has been partially funded by the European Union Horizon-2020 Projects 5G-MoNArch and 5G-Transformer under Grant Agreements 761445 and 761536 as well as by the Network for the Promotion of Young Scientists (TU-Nachwuchsring), TU Kaiserslautern with individual funding.

References

[1] GSMA, "An introduction to network slicing," 2017.

[2] "5G network slicing for cross industry digitization: Position paper," https://www.fokus.fraunhofer.de/download.5G-Network-Slicing_whitepaper.pdf.

[3] K. Samdanis, X. Costa-Perez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, 2016.

[4] C. Marquez, M. Gramaglia, M. Fiore et al., "How should I slice my network? A multi-service empirical evaluation of resource sharing efficiency," in *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking (Mobicom)*, 2018.

[5] V. Sciancalepore et al., "Slice as a service (SlaaS): Optimal IoT slice resources orchestration," in *IEEE Global Communications Conference (GLOBECOM)*, Dec 2017, pp. 1–7.

[6] A. E. Kalor, R. Guillaume, J. J. Nielsen, A. Mueller, and P. Popovski, "Network slicing in industry 4.0 applications: Abstraction methods and end-to-end analysis," *IEEE Transactions on Industrial Informatics*, 2018.

[7] B. Han, L. Ji, and H. D. Schotten, "Slice as an evolutionary service: Genetic optimization for inter-slice resource management in 5G networks," *IEEE Access*, vol. 6, no. 1, pp. 33 137–33 147, 2018.

[8] D. Bega, M. Gramaglia *et al.*, "Optimising 5G infrastructure markets: The business of network slicing," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2017.

[9] E. Parzen, *Modern probability theory and its applications*. John Wiley & Sons, Incorporated, 1960.

[10] L. H. Chen, "Poisson approximation for dependent trials," *The Annals of Probability*, pp. 534–545, 1975.

[11] J. F. Shortle, J. M. Thompson, D. Gross, and C. M. Harris, *Fundamentals of Queueing Theory*. John Wiley & Sons, 2018.

[12] J. D. Little, "A proof for the queuing formula: $L = \lambda W$," *Operations Research*, vol. 9, no. 3, pp. 383–387, 1961.

[13] D. G. Kendall, "Some problems in the theory of queues," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 151–185, 1951.

[14] C. Ancker Jr and A. Gafarian, "Some queuing problems with balking and reneging–I," *Operations Research*, vol. 11, no. 1, pp. 88–100, 1963.

[15] ——, "Some queuing problems with balking and reneging–II," *Operations Research*, vol. 11, no. 6, pp. 928–937, 1963.

[16] S. Kullback, *Information Theory and Statistics*. Courier Corp., 1997.

[17] L. Zanzi, V. Sciancalepore et al., "OVNES: Demonstrating 5G network slicing overbooking on real deployments," in *IEEE Conference on Computer Communications Workshops (INFOCOM DEMO)*, April 2018.

[18] JX.Salvat, L. Zanzi, et al., "Overbooking Network Slices through Yield-driven End-to-End Orchestration," in *ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, December 2018.

[19] V. Sciancalepore, K. Samdanis et al., "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE Conference on Computer Communications (INFOCOM)*, 2017.

[20] O. Sallent, J. Perez-Romero, R. Ferrus, and R. Agusti, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communications*, vol. 24, no. 5, pp. 166–174, 2017.

[21] P. L. Vo, M. N. Nguyen, T. A. Le, and N. H. Tran, "Slicing the edge: Resource allocation for ran network slicing," *IEEE Wireless Communications Letters*, 2018.

[22] J. Vilaplana, F. Solsona, I. Teixidó, J. Mateo, F. Abella, and J. Rius, "A queuing theory model for cloud computing," *The Journal of Supercomputing*, vol. 69, no. 1, pp. 492–507, 2014.

[23] X. Chang, B. Wang, J. K. Muppala, and J. Liu, "Modeling active virtual machines on IaaS clouds using an M/G/m/m+ K queue," *IEEE Transactions on Services Computing*, vol. 9, no. 3, pp. 408–420, 2016.

[24] F. Li, J. Cao, X. Wang, and Y. Sun, "A QoS guaranteed technique for cloud applications based on software defined networking," *IEEE Access*, vol. 5, pp. 21 229–21 241, 2017.

[25] M. Guo, Q. Guan, and W. Ke, "Optimal scheduling of VMs in queueing cloud computing systems with a heterogeneous workload," *IEEE Access*, vol. 6, pp. 15 178–15 191, 2018.

[26] S. Bocquet, "Queueing theory with reneging," Defence Science and Technology Organisation, Australia, Tech. Rep., 2005.

[27] De-quan Yue and Yan-ping Sun, "Waiting time of M/M/c/N queuing system with balking, reneging, and multiple synchronous vacations of partial servers," *Systems Engineering-Theory & Practice*, vol. 28, no. 2, pp. 89–97, 2008.