Slicing Cell Resources: The Case of HTC and MTC Coexistence

Vincenzo Mancuso[†] Paolo Castagno[*] Matteo Sereno[*] Marco Ajmone Marsan[‡][†]

1Università di Torino
Turin, Italy

[†]Imdea Networks Institute
Madrid, Spain

[‡]Politecnico di Torino
Turin, Italy

***Abstract*—In this paper we investigate the allocation of resources to slices on the radio interface of one cell. In particular, we develop a detailed stochastic model of the behaviour of the sliced cell radio access, including most features of the standard access procedures. Our model allows the computation of the throughput achieved by each slice, as well as the distribution of delays for each slice. The availability of a model capable of accurately predicting the performance achieved by services using different slices as a function of the cell parameters is extremely important for the automated run time management of the cell and for the correct setting of its parameters.**

Specifically, while our model can cope with a number of slices, we focus on the case of one cell comprising one slice for human type communications and one slice for machine type communications, and we discuss relevant emerging behaviours in the slices performance, as functions of the cell parameters.

We validate the analytical predictions by comparison against the estimates of a detailed simulator, proving the accuracy of the model. Our model turns out to be very effective in providing insight and guidelines for allocation and management of resources in cells hosting slices carrying traffic derived from services with different characteristics and performance requirements.

I. Introduction

Network slicing is a defining feature of the 5G technology. It allows the presence of several tenants on one infrastructure, and the effective coexistence of services with quite different characteristics and requirements in different virtual slices of the same network. The NGMN (Next Generation Mobile Network) Alliance [1], formed by mobile network operators and equipment manufacturers, gives the following definition of Network Slice Instance [2]: "a set of network functions, and resources to run these network functions, forming a complete instantiated logical network to meet certain network characteristics required by the Service Instance(s)." Network slicing is thus based on the allocation of a shared or dedicated portion of the network resources to each slice, to achieve the best possible Quality of Service (QoS) for each slice, expressed by means of the relevant key performance indicators (KPIs) like throughput, latency, service availability, etc.

ETSI Technical Specification 123 501 [3] defines three classes of slices and service types (SST). The first class refers to slices "suitable for the handling of 5G enhanced mobile broadband" (eMBB). The second class refers to slices "suitable for the handling of ultra-reliable low latency communications" (URLLC.) The third class refers to slices "suitable for the handling of massive IoT" (mIoT). Several slices of the same class can coexist on one infrastructure.

The allocation of resources to the individual slices and their real-time management can be implemented with the support of Software Defined Networking (SDN) and Network Function Virtualization (NFV) approaches, hence with management and orchestration (MANO) functions, and in particular with a a resource orchestrator (RO) that is charged with monitoring the KPIs on the different slices and properly managing resources, so as to avoid Service Level Agreement (SLA) violations.

While several papers already looked at the issues related to resource orchestration (as we discuss in the Related Work Section), in this paper we look at the problem of resource allocation to slices on the radio interface of one cell, an issue which, to the best of our knowledge, has not yet been considered in the technical literature. In particular, we develop a detailed stochastic model of the behaviour of the sliced cell radio access, including: i) Access Class Barring (ACB) techniques, ii) Random Access CHannel (RACH) procedures, iii) preamble decoding and Random Access Response (RAR), iv) Radio Resouce Control (RRC) procedures.

The development of a model capable of predicting the QoS achieved by services using the different slices available in the cell as a function of the cell parameters is extremely important for the automated run time management of the cell and for the correct setting of its parameters, aiming at the simultaneous fulfillment of SLAs in all slices.

Our model builds on the approach presented in [4] and extends it to account for the presence of slices. It allows the computation of the throughput achieved by each slice, as well as the distribution of delays for each service in each slice.

We focus in particular on the case of eMBB and URLLC slices. This is because today one of the key questions about 5G KPIs concerns the possibility of coexistence of eMBB for the provision of an increasingly rich gamut of services to human end users (HTC – human-type communications), together with the URLLC required by the machine-type communications (MTC) necessary for the implementation of the smart factory and industry 4.0 concepts, as well as for many emerging applications requiring strict real-time communications.

Our main contributions in this paper are the following.

- We develop a flexible detailed analytical model for the performance analysis of one cell hosting several slices.

- We provide expressions for the computation of relevant KPIs, such as slice throughput and latency distribution.
- We apply the model to the investigation of the performance of one cell hosting one slice of eMBB type and one slice of URLLC type.
- We provide insight and guidelines for the allocation and management of resources in cells hosting eMBB and URLLC slices.

The rest of this paper is organized as follows. Section II positions our work with respect to some relevant previous works. Section III provides a detailed description of the system we consider. Section IV presents our analytical model and derives expressions for the cell KPIs. Section V describes and comments results for the case of one MTC and one HTC slice, validates them by comparison against simulation estimates, and discusses the main insights provided by the model. Finally, Section VI concludes the paper.

II. Related Work

Three large research projects funded by the European Commission in the framework of the Horizon 2020 5G PP are today developing concepts and implementations of network slicing: 5G-CROSSHAUL [5], 5G-TRANSFORMER [6], and 5G-NORMA [7].

An overview of network slicing concepts, architectures and algorithms was recently provided by two special issues of the IEEE Communications Magazine [8], [9].

Performance issues of network slicing were first considered in [10], where a dynamic RAN cell slicing controller was proposed and evaluated by simulation in a urban setting comprising 19 microcells, and showing that the proposed controller performs better than a distributed static slicing solution and a centralized load balancing solution.

Several papers looked at the optimization of network slicing schemes.

- The optimal allocation of resources to slices was addressed in [11], where a distributed algorithm was proposed and analyzed by simulation, considering a dense small cell deployment, and showing that substantial capacity savings can be achieved while providing a given QoS to end users.
- An optimization problem for radio resource sharing among slices in a cell is studied in [12], that also proposes an efficient algorithm for optimization. Simulation results show good isolation and an increase in the multiplexing gain by sharing unused resources.
- The joint optimization of admission control, user association, baseband and radio resource allocation is proposed in [13]. Simulation results show that the proposed scheme achieves better performance than baseline schemes.

A few papers tackled network slicing with game theory approaches.

- The sharing of resources among slices was investigated in [14]. Each slice is assigned a fixed portion of the available resources, which are then equally distributed to slice users. Newly arriving users are accepted by slices with autonomous decisions based on a game that is shown to admit a Nash equilibrium. The effectiveness of the proposed solution is studied by simulation.
- A study of the dynamic allocation of base station resources to network slices is considered in [15]. The selected resource sharing model is a Fisher Market in economics terms. It is shown to provide each slice with the same or better utility than a static resource allocation and to admit a Nash equilibrium. The performance of the proposed approach is again investigated by simulation.
- The analysis of the market composed by one infrastructure provider and several tenants that rent a network slice to provide service to their customers was tackled in [16]. A slice admission control algorithm is designed to maximize the revenues of the infrastructure provider while providing the expected performance to the slice users. The performance of the proposd algorithm is evaluated by simulation.

The introduction of a limit on the number of resource blocks allocated to each slice in a base station (BS) to guarantee resource isolation is proposed in [17]. The authors show that this approach combined with slight modifications of the ordinary packet scheduling algorithm can provide the desired isolation. In some cases an improvement in throughput with respect to a static bandwidth partitioning is observed in simulation results.

Our work is different from the previous literature because we consider for the first time network slicing together with the details of the algorithms that rule the operations on the radio interface of a base station. In addition, our analysis is based on a detailed analytical model of the base station operations, and simulation just serves the purpose of validating the accuracy of the analytical model.

III. Slicing Radio Access Resources

Here we describe the approach we consider for sharing cell resources among slices. Table I shows the notation we use.

A. Access and Connection Procedures

All devices that need to access a service, of both MTC and HTC types must execute the random access procedure, that starts when a RACH (Random Access CHannel) opportunity (RAO) is offered by the BS. Before accessing the RACH, a terminal may be delayed by the ACB (Access Class Barring) procedure, that allows a prioritization in the RACH access. Barring a service request of a service class happens with a given probability, that we call p_A.

The RACH procedure consists in a packet handshake to synchronize BS and terminal and to assign a unique identifier to the terminal request. A request is successful only when resources are actually allocated to the terminal with the signaling messages that are exchanged after the random access success. Indeed, the standard 3GPP access procedure includes the RACH access phase and the RRC (Radio Resource Control) connect phase, with four messages exchanged in total. In case

TABLE I
NOTATION

Notation	*Description*
A	ACB backoff
B	RACH backoff
C	BS capacity
$C^{(i)}$	Capacity dedicated to slice i
C_s	Shared BS capacity
$k_{\max}$	Maximum number of RACH attempts
M	max RRC_CONNECTED terminals
$M^{(i)}$	RRC_CONNECTED terminals guaranteed to slice i
M_s	shared RRC_CONNECTED places
N_p	Number of random access preambles
$o^{(i)}$	Power ramping offset for slice i
p_A	Barring probability
p_B	Network blocking probability
p_C	RACH collision probability
p_R	Probability of RACH failure (beside collisions)
τ	RAO interval
$T_{\min}$	Minimum time needed to get a RACH reply
$T_{\max}$	RACH timeout
T_O	Application timeout
Y_k	time elapsed as the request leaves stage k
Z	time spent in a RACH stage waiting for an ACK
ζ	Exogenous arrivals
λ	RACH request arrivals
Θ	RACH limit per RAO
σ	Flow of acknowledged RACH requests
ϕ	Flow of decoded RACH requests
ψ	RACH throughput
ξ	Network throughput

of failure during one of the two stages, the terminal repeats its attempt after a random backoff delay, possibly with different transmission power, according to the standard power ramping mechanism that defines how nodes progressively increase their transmission power after each failed attempt [18]. Different backoff values can be defined for failures in different points of the procedure.

In the RACH access phase, the terminal chooses one out of N_p available preambles, and transmits it at the next available RAO. If several terminals choose the same preamble, a collision occurs, and the access request cannot be decoded. If just one terminal chooses a given preamble, its request is decoded, provided the terminal transmission power is high enough. If a collision occurs, or the power is too low, the RACH access must be repeated.

If a request is decoded, the terminal can receive an acknowledgment from the BS. There is a limit (denoted by Θ) to the maximum number of ACKs that can be transmitted by the BS for each RAO, so that a decoded request can receive no ACK, if the limit is reached. If no ACK is received, the terminal must repeat the RACH access procedure.

Terminals that complete the access procedure can move to the RRC_CONNECTED state and receive service from the BS. A limit exists to the maximum number of terminals that can be in the RRC_CONNECTED state (we call it M), so that there is a possibility that the terminal request is blocked even after receiving an ACK. In this case, the terminal notifies the user with an error message equivalent to the *busy tone* in the voice phone system.

A maximum number of repetitions for the RACH access

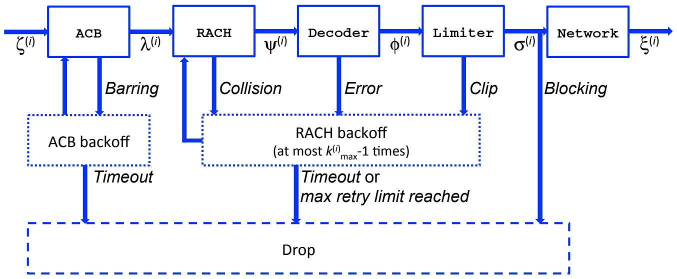

Fig. 1. System blocks representing sliced network functions for the i-th slice hosted by a base station.

procedure is defined, called $k_{\max}$. After $k_{\max}$ attempts, a request is dropped. A repetition can be due to collision (with probability p_C) and to no ACK received (with probability $p_R(k)$ at the k-th attempt, with the associated power level).

In addition, a maximum amount of time is defined for the completion of an access procedure instance. When this time is reached, a timeout expires, and the instance is dropped.

Once a terminal is in the RRC_CONNECTED state, it receives its share of the BS capacity, in terms of allocated resource blocks.

This whole procedure is illustrated in Fig. 1, where we see new service request generation on the left, the `ACB` subsystem, followed by the `RACH`, the `Decoder` and the `Limiter`, all with their backoffs, timeout possibilities and maximum number of retries. The `Network` subsystem corresponds to service by the BS, if no blocking occurs.

In the system, network blocking, timeout and exceeding the RACH retry limit lead to drop the connection attempt.

B. Sliced System

In case of a sliced system, it is necessary to define an allocation of the BS resources to the different slices. We identify the slice parameters by means of a superscript denoting the slice index.

In the spirit of the 3GPP standard, we assume that the barring probability is a characteristic of a service, but since we allocate one service to each slice, the barring probability $P_A^{(i)}$ depends on the slice. The power ramping offset can provide a significant differentiation among slices, increasing the probability of decoding for the slices using higher power. For this reason we will consider different values for different slices. The subset of RACH preambles that can be used by a slice significantly impacts the collision probability. We will thus consider the case of different subsets (possibly with non null intersection) of preambles for different slices. We will instead assume that the ACKs provided by the BS to service requests that succeed on the RACH and at the decoder are equally available to all slices. Obviously, the maximum number of terminals in the RRC_CONNECTED state is a key aspect for governing the slice KPIs, and we will thus consider cases where values are different for different slices. These values also impact the BS bandwidth share obtained by each terminal, since we assume that the BS allocates portions of bandwidth to slices, which are then equally shared by the terminals of that slice which are in the in the RRC_CONNECTED state. Out of the M available positions, we reserve $M^{(i)}$ for unique use of slice i, with the sum of the $M^{(i)}$ less or equal to M. The remaining positions are shared by all slices. The values of backoff delays and access timeouts must be tailored to the types of service and the KPI goals of each slice, so that they must be carefully set by the operator.

IV. Analysis

We model a sliced system that represents the chain that goes from radio connection to network service within a cell. We let out of the analysis the connection from the BS to the core of the network and study in detail BS resource slicing.

A. System flows

The reference system is the one illustrated in Fig. 1, which includes the network functions described in the previous section. As shown in the figure, each block can either promote a connection request to the next level, until service is completed, or yield a failure event. The figure only indicates flows and some configuration parameters for slice i, although we assume the presence of S slices.

`ACB` sees a flow $\zeta^{(i)}$ as input. Failures on the ACB incur backoffs $A^{(i)}$ for slice i, with no limits on the number of back-to-back attempts. We assume that, within a slice, ACB backoffs are i.i.d. and exponentially distributed.

`RACH` receives the flow $\lambda^{(i)}$ from the ACB, which is no higher than $\zeta^{(i)}$ due to the possibility of timeout in ACB. Failures on a RACH access attempt can be due to collision, decoding errors or clipping at `Limiter`. A user cannot distinguish which type of failure occurred, it simply observes that the BS does not acknowledge its request in an interval $T_{\max}$ and then it schedules a RACH backoff before another attempt will start. We call *stage* k the k-th RACH access attempt. We assume that RACH backoffs $B^{(i)}$ are i.i.d. r.v.'s with exponential distribution. Each RACH stage k produces a flow $\psi_k^{(i)}$ of successes, which feeds `Decoder`. Of course, the total flow of successes leaving `RACH` is $\psi^{(i)} = \sum_{k=1}^{k_{\max}^{(i)}} \psi_k^{(i)}$.

`Decoder` introduces losses based on a decoding probability that depends on the RACH stage, because of power ramping (with offset). The output of `Decoder` is a flow $\phi^{(i)} \leq \psi^{(i)}$, which feeds `Limiter`.

`Limiter` causes failures due to the cap Θ on the number of RACH acknowledgments per RAO. This is a hard limit for the ensemble of slices running on the same BS. The output of `Limiter` is a set of flows $\sigma^{(i)}$, one per slice, such that $\sum_{i=1}^{S} \sigma^{(i)} \leq \Theta/\tau$.

If a connection request eventually reaches `Network`, it can still be blocked if the BS network processor has no position left for that slice (and in the shared pool). Blocking happens with probability $p_B^{(i)}$. Conversely, successful requests are served by the network, with a per-slice throughput denoted by $\xi^{(i)} = (1 - p_B^{(i)} \sigma^{(i)})$.

The *busy tone* can therefore be caused by network blocking as well as excessive RACH access attempts (after $k_{\max}^{(i)}$ back-to-back RACH failures) or by specific application timeouts (the app running on the terminal and trying to send a message will not wait forever). The *busy tone* is directly returned to the user as the connection attempt is dropped.

For the framework described above, we now derive expressions for the flows (loads and throughputs) and for the distribution of time spent in the system.

B. Access time

Let us consider a request from slice i that arrives at `ACB`. We denote by $Y_{k-1}^{(i)}$ the time spent by that request from its arrival to ACB to the moment it enters stage k. $Y_{k-1}^{(i)}$ consists of a random number $L^{(i)}$ of barring backoffs, $(k-1)$ times the interval $T_{\max}$ and $k-1$ RACH backoffs.

If there is a success at the $k-th$ stage, the time spent by the request before leaving is $Y_{k-1}^{(i)} + Z$, where the random interval $Z \leq T_{\max}$ is needed to model the delay between RACH request and network grant and it is independent from all r.v.'s $Y_j^{(i)}$, $j = 1, 2, \ldots, k_{\max}^{(i)}$, $i = 1, 2, \ldots, S$. In this case, the request is served with probability $1 - p_B^{(i)}$ or otherwise dropped. Therefore the time spent for a network blocking is same as for a success (because we are not counting the network service in the access time).

If there is a failure due to the maximum number of RACH attempts, the time spent is $Y_{k_{\max}^{(i)}}^{(i)}$ and the request is dropped.

In case of timeout, of course, the time spent is the timeout value selected for slice i, namely $T_O^{(i)}$, and the request is dropped. The distribution of $Y_k^{(i)}$ is

$$F_{Y_k^{(i)}}(x) = \Pr\left\{ L^{(i)} A^{(i)} + k\left(T_{\max} + B^{(i)}\right) \leq x \right\}, \quad (1)$$

where $L^{(i)} \geq 0$ is the random number of back-to-back deferrals experienced because of ACB, due to the barring probability associated to slice i.

Similarly, the distribution of $Y_{k-1}^{(i)} + Z$ is

$$F_{Y_{k-1}^{(i)}+Z}(x) = \Pr\left\{ L^{(i)} A^{(i)} + (k-1)\left(T_{\max} + B^{(i)}\right) + Z \leq x \right\}. \quad (2)$$

Because of the independence of the random variables used in the above expressions, denoting by f_Z the p.d.f. of Z, the following useful result holds:

$$F_{Y_{k-1}^{(i)}+Z} = F_{Y_{k-1}^{(i)}} * f_Z. \quad (3)$$

Moreover, the sum of a fixed numebr of exponential RACH backoffs is an Erlang r.v., and the sum of a geometrically distributed number of ACB exponential backoffs with ACB backoff probabiltity $p_A^{(i)}$ and average ACB backoff $E[A]$ exhibits the following cumulative distribution:

$$\Pr\left\{ L^{(i)} A^{(i)} \leq x \right\} = 1 - p_A^{(i)} e^{-\left(1 - p_A^{(i)}\right) \frac{x}{E[A]}}, \quad \forall x \geq 0. \quad (4)$$

C. RACH stages

A request enters RACH stage 1 if its timeout does not expire during the ACB backoffs. We denote such probability as $P_N^{(i)}(1)$, which is computed through (4) evaluated at $x = T_O^{(i)}$.

Subsequently, and while the timeout does not expire, a request leaves the RACH stage with either a success, or progress to the next stage upon a collision, or a failure in `Decoder` or in `Limiter`. We indicate the probability to access stage k as $P_N^{(i)}(k)$, for which we derive the following recursive expression:

$$P_N^{(i)}(k+1)=P_N^{(i)}(k)\Big[1-\Big(1-p_C^{(i)}\Big)\Big(1-p_R^{(i)}(k)\Big)\Big]F_{Y_k^{(i)}}(T_O^{(i)}). \quad (5)$$

In the above expression, $p_C^{(i)}$ indicates the collision probability in `RACH`, $p_R^{(i)}(k)$ is the probability of failure in either `Decoder` or `Limiter` in stage k, and $F_{Y_k^{(i)}}(T_O^{(i)})$ is the probability that a timeout does not occur before the end of the backoff of stage k. We will derive such quantities later in this section. Before that, we need to derive the general expressions for the probabilities of the following events to occur: excess RACH retries, success, blocking, and timeout. Those events fully characterize the success of the access attempt.

D. Event probabilities

RACH retry limit exceeded. The quantity $P_N^{(i)}\Big(k_{\max}^{(i)}+1\Big)$, formally defined as for other values of k in (5), represents the fraction of $\zeta^{(i)}$ that exceeds the RACH retry limit.

Access attempt success. The fraction of $\zeta^{(i)}$ that observes a success in stage k is derived as the fraction of requests that enters stage k and experiences no failure:

$$P_S^{(i)}(k)=P_N^{(i)}(k)\Big(1-p_C^{(i)}\Big)\Big(1-p_R^{(i)}(k)\Big)\Big(1-p_B^{(i)}\Big)F_{Y_{k-1}^{(i)}+Z}(T_O^{(i)}). \quad (6)$$

The total success probability of slice i, i.e., the fraction of $\zeta(i)$ requests that succeeds, is therefore $P_S^{(i)}=\sum_{k=1}^{k_{\max}^{(i)}}P_S^{(i)}(k)$.

Network blocking. This is similar to the case of success in stage k, but with a network blocking failure:

$$P_B^{(i)}(k)=P_N^{(i)}(k)\Big(1-p_C^{(i)}\Big)\Big(1-p_R^{(i)}(k)\Big)p_B^{(i)}\,F_{Y_{k-1}^{(i)}+Z}(T_O^{(i)}). \quad (7)$$

The fraction of access requests $\zeta^{(i)}$ that experiences network blocking is thus $P_B^{(i)}=\sum_{k=1}^{k_{\max}^{(i)}}P_B^{(i)}(k)$.

Timeout. A timeout can occur either during ACB backoffs, with probability $P_{TO}^{(i)}(0)=1-P_N^{(i)}(1)$, or during RACH operations. In the k-th stage, a fraction of requests suffer a timeout while waiting for the network grant or during the backoff. Hence, for $k\geq 1$:

$$\begin{aligned}P_{TO}^{(i)}(k)=P_N^{(i)}(k)\Big\{&\Big(1-p_C^{(i)}\Big)\Big(1-p_R^{(i)}(k)\Big)\Big[1-F_{Y_{k-1}^{(i)}+Z}(T_O^{(i)})\Big]\\ &+\Big[1-\Big(1-p_C^{(i)}\Big)\Big(1-p_R^{(i)}(k)\Big)\Big]\Big[1-F_{Y_k^{(i)}}(T_O^{(i)})\Big]\Big\}. \quad (8)\end{aligned}$$

The total timeout probability observed by a slice is therefore

$$P_{TO}^{(i)}=\sum_{k=0}^{k_{\max}^{(i)}}P_{TO}^{(i)}(k); \quad (9)$$

Busy tone. Access requests that exceed the RACH retry limit, experience a network blocking event or a timeout are dropped. Therefore, the busy tone is sent with probability $1-P_S^{(i)}=P_N^{(i)}\Big(k_{\max}^{(i)}+1\Big)+P_B^{(i)}+P_{TO}^{(i)}$.

E. Derivation of throughputs and loads with cycles

With the expressions derived so far, we have characterized the trajectory of the exogenous access requests that feed the system for slice i, i.e., $\zeta^{(i)}$. However, the expressions derived are functions of three parameters that we need to derive next: $p_C^{(i)}$, $p_R^{(i)}(k)$, and $p_B^{(i)}$.

RACH collision probability and throughput. The input of `RACH` is the flow $\lambda^{(i)}$ that arrives from `ACB`. However, `RACH` has internal cycles, and $\lambda^{(i)}$ is just the input to the first stage. With the definitions of Section IV-C, we have the following input flows for each successive stage (note that $\lambda_1^{(i)}=\lambda^{(i)}$):

$$\lambda_k^{(i)}=\zeta^{(i)}P_N^{(i)}(k),\quad k=1,2,\dots k_{\max}^{(i)}. \quad (10)$$

We model `RACH` as a slotted Aloha system with multiple channels. The load of the system is the sum of the requests arriving to the various stages, whereas the number of channels is the number of preambles assigned by the BS to the slice.

Specifically, each slice receives a set of $N^{(i)}$ dedicated preambles. In addition, the BS keeps a pool of N_s shared preambles that can be accessed by all slices. The total number of preambles is $N_p=N_s+\sum_{i=1}^{S}N^{(i)}$.

In each RACH attempt, according to the standard, a terminal selects a preamble uniformly at random, so that the per-preamble RACH load generated by slice i is

$$\ell^{(i)}=\frac{\zeta^{(i)}}{N^{(i)}+N_s}\sum_{k=1}^{k_{\max}^{(i)}}P_N^{(i)}(k). \quad (11)$$

The collision probability over a single preamble j, from slotted Aloha results with slots of duration τ, is as follows:

$$p_{C,j}=\begin{cases}1-e^{-\tau\ell^{(i)}},\ 1\leq i\leq S, & \text{dedicated preamble;}\\ 1-e^{-\tau\sum_{i=1}^{S}\ell^{(i)}}, & \text{shared preamble.}\end{cases} \quad (12)$$

The resulting per-slice RACH collision probability is derived as average of (12) over the preambles used by a slice and selected uniformly at random at each attempt:

$$p_C^{(i)}=1-\frac{N^{(i)}e^{-\tau\ell^{(i)}}+N_s\,e^{-\tau\sum_{q=1}^{S}\ell^{(q)}}}{N^{(i)}+N_s}. \quad (13)$$

The throughput of `RACH` (for slice i and stage k) is:

$$\psi_k^{(i)}=\Big(1-p_C^{(i)}\Big)\lambda_k^{(i)},\quad \psi^{(i)}=\sum_{k=1}^{k_{\max}^{(i)}}\psi_k^{(i)}. \quad (14)$$

Throughput of `Decoder`. At each stage of the RACH, `Decoder` has a different failure probability, due to power ramping in RACH message transmissions [18]. `Decoder` failure probability is expressed as $e^{-k-o^{(i)}}$, where k is the RACH attempt stage and $o^{(i)}$ is the slice offset. Therefore, at stage k, slice i observes the following `Decoder` throughput:

$$\phi_k^{(i)}=\psi_k^{(i)}\Big(1-e^{-\big(k+o^{(i)}\big)}\Big) \quad (15)$$

which sums up to a flow $\phi^{(i)}=\sum_{k=1}^{k_{\max}^{(i)}}\phi_k^{(i)}$.

Losses due to `Limiter`. The BS can only grant Θ requests per RAO, shared between the slices. Therefore there are losses when the output of `Decoder` in a RAO interval is higher than Θ requests.

With the RACH preamble partition described above, we can compute the distribution of successes per RAO and hence compute the average loss due to `Limiter`.

In a pool of W preambles subject to homogeneous per-preamble load—e.g., in a pool of shared preambles, or in a pool of preambles dedicated to a single slice—the probability ω_a to have exactly a decoded messages in a RAO is approximated with the probability of having a successes over W i.i.d. Bernoulli experiments (one per RACH preamble, which can only output no or one decoded request). The success probability of each Bernoulli experiment is computed from the aggregate number of messages decoded over those preambles in an interval τ, as shown next.

For a pool of dedicated preambles $N^{(i)}$, the collision probability is the same for all preambles and it is given by (12). For each preamble used, the average output in a RAO, after the decoding, is therefore

$$p^{(i)} = \tau e^{-\tau \ell^{(i)}} \sum_{k=1}^{k_{\max}^{(i)}} \left(1 - e^{-\left(k+o^{(i)}\right)}\right) \frac{\lambda_k^{(i)}}{N^{(i)} + N_s}, \qquad (16)$$

which can be regarded as the Bernoulli success probability of dedicated preambles. For the shared pool, the result is similar:

$$p_s = \tau e^{-\tau \sum_{i=1}^{S} \ell^{(i)}} \sum_{i=1}^{S} \sum_{k=1}^{k_{\max}^{(i)}} \left(1 - e^{-\left(k+o^{(i)}\right)}\right) \frac{\lambda_k^{(i)}}{N^{(i)} + N_s}. \qquad (17)$$

For a dedicated pool we have the following distribution:

$$\omega_a^{(i)} = \begin{cases} \binom{N^{(i)}}{a} \left(p^{(i)}\right)^a \left(1-p^{(i)}\right)^{N^{(i)}-a}, & a \in \{0,..,N^{(i)}\}; \\ 0, & \text{otherwise}; \end{cases} \qquad (18)$$

while for the shared pool of preambles:

$$\omega_{sa} = \begin{cases} \binom{N_s}{a} (p_s)^a (1-p_s)^{N_s-a}, & a \in \{0,..,N_s\} \\ 0, & \text{otherwise.} \end{cases} \qquad (19)$$

Finally, putting together the different pools, the probability Ω_a to have exactly a messages decoded (from any slice) is

$$\Omega_a = \sum_{a_1=0}^{N_p} \sum_{a_2=0}^{N_p} \cdots \sum_{a_S=0}^{N_p} \omega_a^{(1)} \omega_a^{(2)} \ldots \omega_a^{(S)} \omega_{s\left(a-\sum_{r=1}^{S} a^{(r)}\right)}. \qquad (20)$$

Overall, the average losses are

$$E\left[N_L\right] = \sum_{a=\Theta+1}^{N_p} (a - \Theta)\, \Omega_a, \qquad (21)$$

and we assume that the losses are spread over the slices proportionally to their load at `Limiter`:

$$E\left[N_L^{(i)}\right] = E\left[N_L\right] \frac{\phi^{(i)}}{\sum_{q=1}^{S} \phi^{(q)}}; \qquad (22)$$

The resulting per-slice `Limiter` throughput is

$$\sigma^{(i)} = \phi^{(i)} - \frac{E\left[N_L^{(i)}\right]}{\tau} = \phi^{(i)} \left(1 - \frac{E\left[N_L\right]}{\tau \sum_{q=1}^{S} \phi^{(q)}}\right). \qquad (23)$$

Since losses at `Limiter` do not discriminate between RACH stages, `Limiter` throughput per-stage, $\sigma_k^{(i)}$, is obtained by replacing $\phi_k^{(i)}$ for $\phi^{(i)}$ in (23).

Computation of $p_R^{(i)}(k)$. This quantity is the aggregate loss rate due to the combined action of `Decoder` and `Limiter` for requests at stage k:

$$p_R^{(i)}(k) = 1 - \frac{\sigma_k^{(i)}}{\psi_k^{(i)}} = 1 - \left(1 - e^{-\left(k+o^{(i)}\right)}\right)\left(1 - \frac{E\left[N_L^{(i)}\right]}{\tau \phi^{(i)}}\right). \qquad (24)$$

Blocking probability. The BS network processor can serve at most M users at the same time. Moreover, each slice disposes of $M^{(i)}$ dedicated places in the BS, and some additional M_s places can be shared among all slices. Arrival in excess are dropped, thus originating a network blocking probability.

The network processor has a total capacity C, out of which $C^{(i)}$ is reserved for slice i, and C_s is shared among slices. If there are up to $M^{(i)}$ users for slice i, they equally share $C^{(i)}$. However, when there are $m^{(i)} > M^{(i)}$ users, $M^{(i)}$ of them get a service rate $\frac{C^{(i)}}{M^{(i)}}$ and the remaining $m^{(i)} - M^{(i)}$ users access shared resources without any priority. Hence their service depends on the total number of users in all the slices, i.e., their serving rate is $C_s / \sum_{i=1}^{S} \min\left(0, m^{(i)} - M^{(i)}\right)$. Moreover, if a job using dedicated resources leaves the system, then a job using shared resources is promoted to use dedicated resources.

The above description reminds of the operation of a multi-class processor sharing (PS) queue in which one class receives part of what cannot be accommodated in the other classes, and jobs can be shuffled and promoted.

Thus, we model the network processor with a PS with S classes and hard limits on the number of customers in service given by $M^{(i)} + M_s$ for each class, with a global limit at M. The capacities of such classes are their dedicated capacity $C^{(i)}$ plus a portion of the shared capacity C_s. The intensities of the arrival rates are the values of the $\sigma^{(i)}$, but shared resources receive the overflow of each class arrivals, when all dedicated resources are busy.

We study the operation of the described PS queue by means a continuous-time Markov chain with S-dimensional state space (one dimension per class, to count the number of jobs) whose transitions are depicted in Fig. 2 for the case of two slices ($S = 2$). In the figure, each state of the chain reports the number of jobs $m^{(1)} \leq M - M^{(2)}$ in slice 1 and $m^{(2)} \leq M - M^{(1)}$ in slice 2, subject to the constraints that $m^{(1)} + m^{(2)} \leq M$. The chain has a precise symmetry and a trapezoidal shape, which is due to the above mentioned constraints.

If we denote by (a, b) the state of the chain, where a is the number of users in slice 1 and b is the number in slice 2, the transition rates for Fig. 2 from state (a, b) to other states are

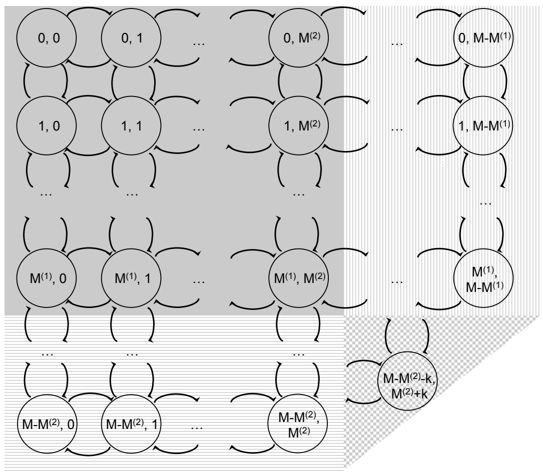

Fig. 2. CTMC describing the Network processor operation for $S = 2$. Top-to-bottom transitions have rate $\sigma^{(1)}$, while left-to-right transitions have rate $\sigma^{(2)}$. Bottom-to-top transitions have rate $C^{(1)}$ in the first $M^{(1)}$ rows and $C^{(1)} + Q^{(1)}(a, b)$ in the remaining rows. Right-to-left transitions have rate $C^{(2)}$ in the first $M^{(2)}$ columns and $C^{(2)} + Q^{(2)}(a, b)$ otherwise.

as follows:
$(a{+}1, b) : \sigma^{(1)}\ \ \forall a \le M^{(1)}{-}1,\ b \le M^{(2)}$and $\forall a \le M - M^{(2)}{-}b{-}1$
$(a, b{+}1) : \sigma^{(2)}\ \ \forall b \le M^{(2)}{-}1,\ a \le M^{(1)}$and $\forall b \le M - M^{(1)}{-}a{-}1$
$(a - 1, b) : C^{(1)}\ \ \forall a \le M^{(1)}$
$(a - 1, b) : C^{(1)} + Q^{(1)}(a, b)\ \ \forall M^{(1)} < a \le M - M^{(2)}$
$(a, b - 1) : C^{(2)}\ \ \forall b \le M^{(2)}$
$(a, b - 1) : C^{(2)} + Q^{(2)}(a, b)\ \ \forall M^{(2)} < b \le M - M^{(1)}$

For simplicity of notation, we used the following quantities:

$$Q^{(1)}(a, b) = \begin{cases} \frac{a - M^{(1)}}{a - M^{(1)} + \max\left(0, b - M^{(2)}\right)} & \text{if } a > M^{(1)}; \\ 0 & \text{otherwise;} \end{cases} \tag{25}$$

$$Q^{(2)}(a, b) = \begin{cases} \frac{b - M^{(2)}}{b - M^{(2)} + \max\left(0, a - M^{(1)}\right)} & \text{if } b > M^{(2)}; \\ 0 & \text{otherwise.} \end{cases} \tag{26}$$

The solution of the Markov chain can be obtained numerically with specialized tools like SMART [19], which can solve chains with tens of thousands of states in a few seconds using normal desktop computers.

We therefore solve the Markov chain numerically to compute $p_B^{(i)}$ as the sum of the relevant state probabilities. I.e., for slice 1 we sum over states that lay on the bottom edge of Fig. 2, whereas for slice 2 on the edge on the right. Note that the diagonal edge in the triangular-shaped part at the right-bottom part of the chain in Fig. 2 contains states in which both slices suffer blocking.

F. Access time distributions

The cumulative distribution of the time $T^{(i)}$ spent in one access attempt in slice i is the one resulting from the following events that partition the space of probabilities: timeout, success or network blocking in stage k, and excess RACH retries. In case of timeout, the time spent is $T_O^{(i)}$. In case of success or network blocking in stage k, the time spent in the access attempt is the r.v. $Y_{k-1}^{(i)} + Z$, conditional to the event that the timeout does not expire. In case of excess RACH retries, the time is $T_{\max}$ (for the last retry with no BS answer) plus the r.v. $Y_{k_{max}-1}^{(i)}$ (which accounts for previous retries), conditional to the event that the time at the end of the last but one attempt allows for an extra $T_{\max}$ in the last attempt. The result is as follows:

$$F_{T^{(i)}}(x) = P_{TO}^{(i)}\, \mathcal{U}\left(x - T_O^{(i)}\right) + \sum_{k=1}^{k_{\max}^{(i)}} \frac{F_{Y_{k-1}^{(i)}+Z}(x)}{F_{Y_{k-1}^{(i)}+Z}\left(T_O^{(i)}\right)} \Big(P_S^{(i)}(k) + P_B^{(i)}(k)\Big) + P_N^{(i)}\left(k_{\max}^{(i)} + 1\right) \frac{F_{Y^{(i)}_{k_{\max}^{(i)}-1}}(x - T_{\max})}{F_{Y^{(i)}_{k_{\max}^{(i)}-1}}\left(T_O^{(i)} - T_{\max}\right)}, \tag{27}$$

where $\mathcal{U}\left(x - T_O^{(i)}\right)$ is a unit step at time $T_O^{(i)}$.

As a corollary of (27), note that the CDF of the time spent in one attempt in case of success is

$$F_{T^{(i)}}(x|\text{Success}) = \frac{1}{P_S^{(i)}} \sum_{k=1}^{k_{\max}^{(i)}} \frac{F_{Y_{k-1}^{(i)}+Z}(x)}{F_{Y_{k-1}^{(i)}+Z}\left(T_O^{(i)}\right)} P_S^{(i)}(k). \tag{28}$$

V. Numerical Results

In this section we study a few cases of sliced BS resources, which correspond to realistic application scenarios. In all cases we consider a BS with user plane capacity equal to 100 Mb/s that must transmit packets of HTC type with average length 1.2 Mb, and with average length 8 kb in case of MTC. In all cases the number of RACH preambles is equal to 54, and the number of positions in the RRC_CONNECT state is 200.

In Table II we provide for the considered scenarios the traffic shares of the MTC and HTC slices, the shares of the BS capacity dedicated to the two slices, the numbers of dedicated RACH preambles and positions in the RRC_CONNECTED state, and the timeout values in seconds. The resources which are not dedicated to slices can be evenly shared.

The four scenarios that we consider are the following.

- **Sparse IoT** – A BS serving a urban cell with mostly HTC traffic, and a small slice for IoT (MTC) traffic.
- **Dense IoT** – A BS serving a cell with mostly MTC connections with low traffic, although a large part of the capacity is used by a few HTC devices.
- **Small Factory** – A BS serving an urban area with mostly HTC traffic, but devoting a slice to serve a urban industrial settlement, with MTC traffic.
- **Big Factory** – A BS serving a private area (such as a smart factory) with mostly MTC traffic, and a slice to handle HTC traffic.

A. Validation

In order to validate the analytical model, we used an ad hoc simulator written in C++. This is an event based simulator that represents with high accuracy the standard operations necessary to register the terminal at the BS, and to access and use the BS resources. The fact that the simulator closely follows the standard 3GPP procedures allows us to validate the

TABLE II
SLICE PARAMETERS FOR THE CONSIDERED APPLICATION SCENARIOS

SCENARIO	TRAFFIC SHARE		CAPACITY SHARE		DEDICATED PREAMBLES		DEDICATED POSITIONS		TIMEOUT [s]	
	HTC	MTC	HTC	MTC	HTC	MTC	HTC	MTC	HTC	MTC
SPARSE IOT	0.95	0.05	0.8	0.02	40	5	100	10	5	5
DENSE IOT	0.05	0.95	0.75	0.05	10	40	40	100	3	1
SMALL FACTORY	0.75	0.25	0.5	0.2	30	10	50	50	5	0.1
BIG FACTORY	0.3	0.7	0.1	0.5	10	30	20	150	5	0.1

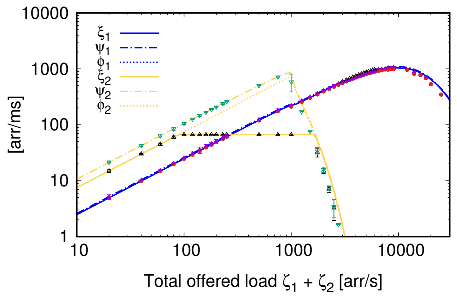

Fig. 3. Small Factory throughput with comparison with simulator

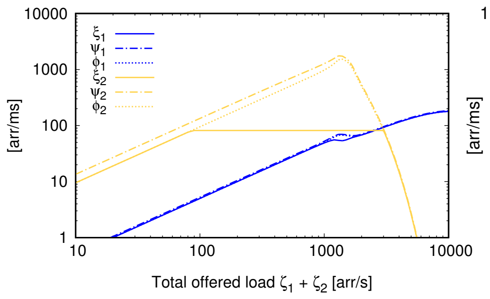

Fig. 4. Sparse IoT throughput

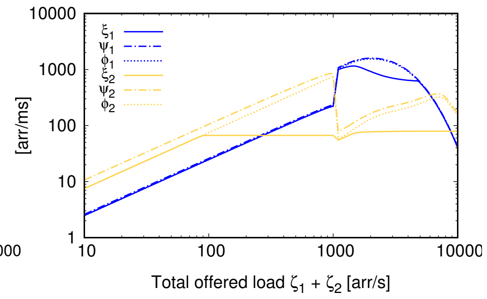

Fig. 5. Dense IoT throughput

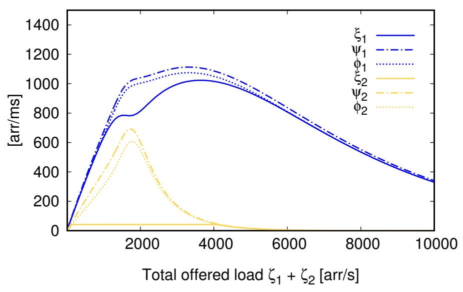

Fig. 6. Big factory Throughput

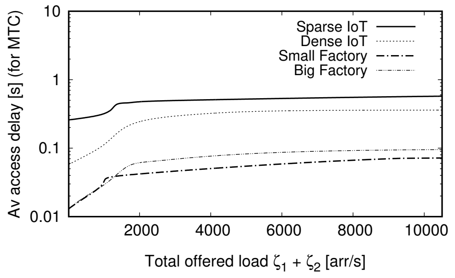

Fig. 7. Average access delays for slice 1 (MTC) for the four different scenarios

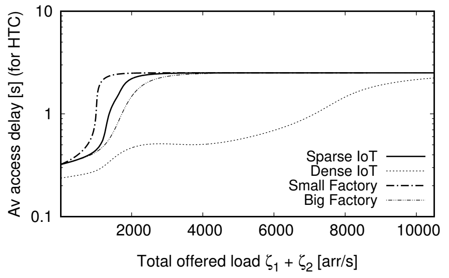

Fig. 8. Average access delays for slice 2 (HTC) for the four different scenarios

simplifying assumptions introduced in the analytical model for the sake of tractability.

Fig. 3 shows the excellent match between simulation and analytical results in the case of the Small Factory scenario. Simulation results are presented together with their 95% confidence intervals.

B. Throughput

Figs. 3 to 6 illustrate the behavior of the system throughput (at `RACH`, `Decoder` and `Network`) for the four considered scenarios. In the Sparse IoT case, HTC saturates first, and the HTC load on the RACH has only a minor impact on the traffic of MTC. This indicates that light MTC traffic with non-stringent delay requirements is not hard to accommodate. The Dense IoT case is more interesting. It shows that MTC and HTC saturation regions superpose. Here, the activity of MTC on the RACH heavily affects HTC performance. Therefore, supporting the coexistence of HTC and MTC slices in such scenarios is challenging. If we go back to the Small Factory case used above for validation, we notice only minor differences with the Sparse IoT case, although here the limited resources allocated to HTC make is easier to avoid HTC interfere and impair MTC KPIs.

More critical is the Big Factory case, in which the MTC traffic is predominant and yet a small amount of HTC connections can seriously hinder MTC performance at relatively low aggregate traffic rates, while under heavy traffic the impact of HTC on the throughput of MTC is less relevant.

The figures also show that the loss due to `Decoder` are negligible for MTC, while they have to be taken into account for HTC. This is due to the fact that we have set a power ramping offset for MTC ($o^{(1)} = 2$) while the HTC does not use any offset. This tells of the importance of the power ramping offset in the slice configuration.

C. Access Delay

The numerical results of access delay for MTC and HTC traffic are shown in Figs. 7 and 8, respectively. In the case of MTC the two curves for Small Factory and Big Factory saturate at 100 ms, which is the timeout for those cases. The other two cases remain well below their timeout values which are much less stringent. In the case of HTC, we see that all curves saturate at the same value, which is close to 2.25 s, due to the maximum permitted number of retries, and the average backoff delay equal to 0.25 s. To this we must add $10T_{\max}$, which is however just about 0.13 s. The Small Factory scenario saturates first because a large fraction of the BS traffic is associated with only a small portion of dedicated resources. The Dense IoT scenario yields the lowest delays because its traffic share is very low, and the reserved resources prove to be sufficient to achieve low delay.

D. Success probability

Success probabilities for the 4 considered scenarios are presented in Figs. 9 and 10 for MTC and HTC traffic, respectively. In the MTC case, Small Factory and Big Factory suffer from the very low timeout values, but achieve good success probabilities up to about 1000 requests/s. Beyond this value, the RACH approaches saturation, and retrials make timeouts more likely. In the case of HTC, we see very high success rates in th case of the Dense IoT scenario, which are due to the fact that the HTC traffic share in this case is very low, and resources reserved to HTC are largely sufficient.

E. Lesson Learnt

One of our key observations is that the saturation of the RACH is a critical issue, and unexpected behaviors are observed for the traffic loads that bring the RACH to saturation. In Fig. 11 we plot the probability of reaching the timeout for MTC traffic versus the HTC traffic load in the Big Factory scenario, assuming that the MTC traffic is fixed at 1000 arrivals/s, and that the number of preambles reserved for MTC is varied between 20 and 40. We clearly see a bump in the timeout probability that corresponds HTC traffic values that lead to RACH saturation. After this point the HTC Traffic consumes little network processor resources, but saturates the RACH, so that it is necessary to protect the MTC traffic by allocating a large number of dedicated preambles. If the number of dedicated preambles is too small, the timeout probability settles at unacceptable values.

VI. Conclusions

In this paper we have described a detailed stochastic model of the behavior of radio access in a sliced cell, including most features of the standard access procedures. Our model allows the investigation of the effect of the allocation of resources to slices on the radio interface of one cell, hence the correct setting of the slice parameters.

Looking at the case of one cell comprising one HTC and one MTC slice, we observed the mutual effects of slice traffic increases on performance, exposing unexpected behaviors close to the traffic values where the RACH is close to saturation.

References

[1] "The NGMN alliance - at a Glance," http://www.ngmn.org, 2011.

[2] The NGMN Alliance. (2016, Jan.) Description of Network Slicing Concept. [Online]. Available: https://www.ngmn.org/uploads/media/160113_Network_Slicing_v1_0.pdf

[3] "System Architecture for the 5G System," 3GPP TS 23.501 Version 15.2.0 - Release 15, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf

[4] Reference blinded for double-blind submission.

[5] X. Li *et al.*, "5G-Crosshaul Network Slicing: Enabling Multi-Tenancy in Mobile Transport Networks," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 128–137, 2017.

[6] C. Casetti *et al.*, "Network slices for vertical industries," in *Proc. of IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, 2018, pp. 254–259.

[7] F. Z. Yousaf *et al.*, "Network slicing with flexible mobility and QoS/QoE support for 5G Networks," in *Proc. of IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017, pp. 1195–1201.

[8] K. Samdanis *et al.*, "5G Network Slicing - Part 1: Concepts, Principles, and Architectures," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 70–71, 2017.

[9] ——, "5G Network Slicing - Part 2: Algorithms and Practice," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 110–111, 2017.

[10] P. C. Garces *et al.*, "RMSC: A Cell Slicing Controller for Virtualized Multi-Tenant Mobile Networks," in *Proc. of IEEE 81st Vehicular Technology Conference (VTC Spring)*, 2015, pp. 1–6.

[11] P. Caballero *et al.*, "Multi-Tenant Radio Access Network Slicing: Statistical Multiplexing of Spatial Loads," *IEEE/ACM Transactions on Networking*, vol. 25, no. 5, pp. 3044–3058, 2017.

[12] C. Y. Chang, N. Nikaein, and T. Spyropoulos, "Radio access network resource slicing for flexible service execution," in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2018, pp. 668–673.

[13] Y. L. Lee *et al.*, "Dynamic Network Slicing for Multitenant Heterogeneous Cloud Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 4, pp. 2146–2161, 2018.

[14] J. Zheng *et al.*, "Statistical multiplexing and traffic shaping games for network slicing," in *15th International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt)*, 2017, pp. 1–8.

[15] P. Caballero *et al.*, "Network slicing games: Enabling customization in multi-tenant networks," in *in Proc. of IEEE INFOCOM*, 2017.

[16] D. Bega *et al.*, "Optimising 5G infrastructure markets: The business of network slicing," in *in Proc. of IEEE INFOCOM*, 2017.

[17] D. Nojima *et al.*, "Resource Isolation in RAN Part While Utilizing Ordinary Scheduling Algorithm for Network Slicing," in *Proc. of IEEE 87st Vehicular Technology Conference (VTC Spring)*, 2018, pp. 1–6.

[18] "Technical Specification Group Radio Access Network; Study on RAN Improvements for Machine-type Communications," 3GPP, TR 37.868 Release 11, 2011.

[19] G. Ciardo, R. L. Jones, A. S. Miner, and R. Siminiceanu, "Logical and stochastic modeling with smart," in *Computer Performance Evaluation. Modelling Techniques and Tools*, P. Kemper and W. H. Sanders, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 78–97.

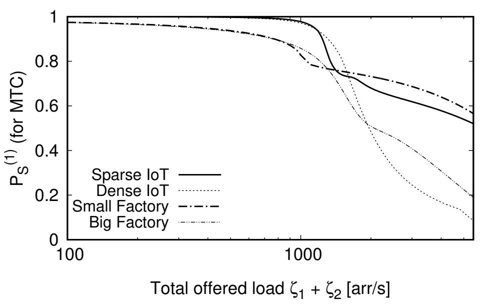

Fig. 9. Success probability for MTC

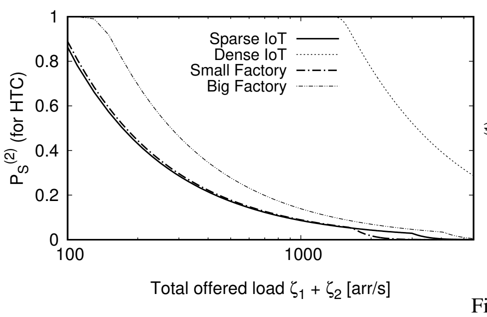

Fig. 10. Success probability for HTC

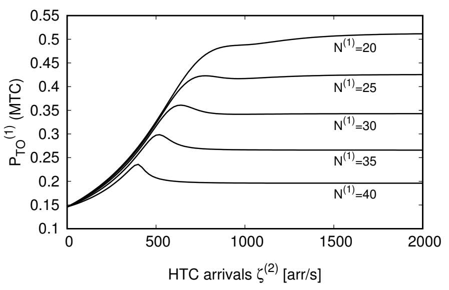

Fig. 11. Timeout probability for MTC at fixed MTC arrival rate $\zeta^{(1)} = 1000$ arr/s.