

This is a postprint version of the following document: (In Press)

Mancuso, V., Castagno, P., Sereno, M. y Marsan, M.A. (2020).
Modeling MTC and HTC Radio Access in a Sliced 5G Base Station.
IEEE Transactions on Network and Service Management.

DOI: <https://doi.org/10.1109/TNSM.2020.3034688>

Modeling MTC and HTC Radio Access in a Sliced 5G Base Station

Vincenzo Mancuso, Paolo Castagno, Matteo Sereno, and Marco Ajmone Marsan, *Life Fellow, IEEE*

Abstract—In this article, we develop a modeling framework to describe the uplink behavior of radio access in a sliced cell, including most features of the standard 3GPP multiple access procedures. Our model allows evaluating throughput and latency of each slice, as a function of cell parameters, when resources are in part dedicated to individual slices and in part shared. The availability of an accurate model is extremely important for the automated run time management of the cell and for the correct setting of its parameters. Indeed, our model considers most details of the behavior of sliced 5G cells, including Access Class Barring (ACB) and Random Access CHannel (RACH) procedures, preamble decoding, Random Access Response (RAR), and Radio Resource Control (RRC) procedures. To cope with a number of slices devoted to serve various co-deployed tenants, we derive a multi-class queueing model of the network processor. We then present (i) an accurate and computationally efficient technique to derive the performance measures of interest using continuous-time Markov chains, which scales up to a few slices only, and (ii) tight performance bounds, which are useful to tackle the case of more than a fistful of slices. We prove the accuracy of the model by comparison against a detailed simulator. Eventually, with our performance evaluation study, we show that our model is very effective in providing insight and guidelines for allocation and management of resources in cells hosting slices for services with different characteristics and performance requirements, such as machine type communications and human type communications.

Index Terms—Radio access network, 5G, base station, slicing, queueing networks, HTC and MTC coexistence.

Manuscript received July 4, 2020; revised October 16, 2020; accepted October 21, 2020. This work is partially supported by the Region of Madrid through the TAPIR-CM project (S2018/TCS-4496), by a Ramon y Cajal grant (ref: RYC-2014-16285) from the Spanish Ministry of Economy and Competitiveness, and by HOME (Hierarchical Open Manufacturing Europe) project supported by the Regione Piemonte, Italia (framework program POR FESR 14/20). This work was supported by the EU Commission through the 5GROWTH project (grant agreement no. 856709). A preliminary version of this work appeared in the proceedings of IEEE Infocom in 2019 [1]. The associate editor coordinating the review of this article and approving it for publication was A. Detti. (*Corresponding author: Vincenzo Mancuso.*)

Vincenzo Mancuso is with the Opportunistic Architectures Lab, IMDEA Networks Institute, 28918 Leganes, Spain (e-mail: vincenzo.mancuso@imdea.org).

Paolo Castagno and Matteo Sereno are with the Computer Science Department, University of Turin, 10124 Torino, Italy (e-mail: paolo.castagno@unito.it; matteo.sereno@unito.it).

Marco Ajmone Marsan is with the Opportunistic Architectures Lab, IMDEA Networks Institute, 28918 Leganes, Spain, and also with the Department of Electronic and Telecommunication, Polytechnic University of Turin, 10138 Torino, Italy (e-mail: ajmone@polito.it).

Digital Object Identifier 10.1109/TNSM.2020.3034688

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 [2], formed by mobile network operators and equipment manufacturers, gives the following definition of *network slice instance* [3]: “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 [4] defines three classes of slices. 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 resource orchestrator that monitors KPIs on different slices and properly manages 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 article we look at the problem of uplink radio 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.¹ In particular, we develop a detailed stochastic model of the behavior 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), and (iv) Radio Resource Control (RRC) procedures.

¹Except for a preliminary version of this work that appeared in [1].

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 we presented in [5] 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 HTC (human-type communications) and MTC (machine-type communications), the former including broadband services, while the latter embraces both time-critical and massive-type services. In both cases we rely on eMBB slices, which are the only suitable type of slice when it comes to handle large numbers of devices active at the same time [4]. Alternative access schemes, e.g., NOMA-based grant-free access techniques are convenient only for devices generating traffic sporadically [6], while fast uplink grants proactively generated by the base station incur non-negligible delay and packet dropping, and reduce the capacity of the cell [7]. This explains why 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, together with the services required by either massive or critical MTC necessary for implementing smart factory, industry 4.0 and IoT concepts.

Our main contributions are as follows:

- 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 up to six slices for HTC and MTC.
- We provide insight and guidelines for the allocation and management of resources in cells hosting HTC and MTC slices.

The rest of this article is organized as follows. Section II provides a detailed description of the studied system. Section III presents our analytical model and derives expressions for KPIs. Section IV describes and comments results for the case of two to six MTC/HTC slices, validates them by comparison against simulation, and discusses the main model's insights. Section V positions our work with respect to previous work. Finally, Section VI concludes this article.

II. SLICING RADIO ACCESS RESOURCES

Here we describe radio resource sharing among slices, using the notation of Table I.

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)

TABLE I
NOTATION USED IN THIS ARTICLE

<i>Notation Description</i>			
$A^{(i)}$	ACB backoff for slice i	$p_R^{(i)}$	RACH failure prob. without collision (slice i)
$B^{(i)}$	RACH backoff for slice i	τ	RAO interval
C	BS capacity	T_{\min}	Minimum time needed to get a RACH reply
$C^{(i)}$	BS capacity dedicated to slice i	T_{\max}	RACH timeout
C_s	shared BS capacity	$T_O^{(i)}$	application timeout for slice i
$k_{\max}^{(i)}$	max number of RACH attempts (slice i)	$Y_k^{(i)}$	time elapsed when leaving stage k (slice i)
M	max number of RRC_CONNECTED terminals	Z	time spent in a RACH stage waiting for an ACK
$M^{(i)}$	guaranteed RRC_CONNECTED positions (slice i)	$\zeta^{(i)}$	exogenous arrivals for slice i
M_s	shared RRC_CONNECTED places	$\lambda^{(i)}$	RACH request arrivals for slice i
N_p	number of random access preambles	Θ	RACH limit per RAO (slice-oblivious)
$o^{(i)}$	power ramping offset for slice i	$\sigma^{(i)}$	flow of acknowledged RACH requests (slice i)
$p_A^{(i)}$	barring probability for slice i	$\phi^{(i)}$	flow of decoded RACH requests for slice i
p_B	blocking probability (same for all slices)	$\psi^{(i)}$	RACH throughput for slice i
$p_C^{(i)}$	RACH collision probability observed by slice i	$\xi^{(i)}$	Network processor throughput for slice i

procedure, that allows a prioritization in the RACH access. Barring a service request of a service class happens with a given probability.

The RACH procedure consists in a packet handshake to synchronize BS and terminal and to assign a unique identifier to the terminal service 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 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 3GPP power ramping mechanism that defines how nodes progressively increase their transmission power after each failed attempt [8]. 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. Note that this is a conservative assumption, since decoding could happen for the highest power transmission in some cases, due

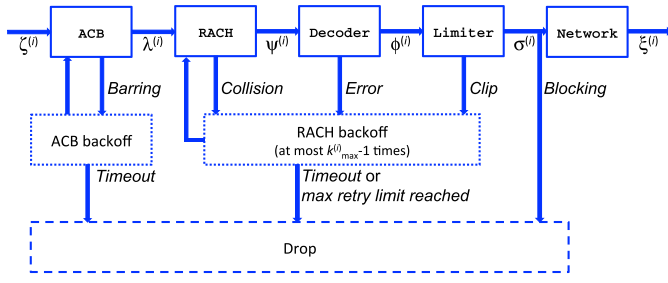


Fig. 1. System blocks representing sliced network functions for the i -th slice hosted by a base station.

155 to the radio capture effect; our analysis is thus slightly pes-
 156 simistic. If just one terminal chooses a given preamble, its
 157 request is decoded, provided the terminal transmission power
 158 is high enough. If a collision occurs, or the power is too low,
 159 the RACH access must be repeated.

160 If a request is decoded, the terminal can receive an acknowl-
 161 edgment from the BS. There is a limit (denoted by Θ in this
 162 article) to the maximum number of ACKs that can be trans-
 163 mitted by the BS for each RAO, so that a decoded request can
 164 receive no ACK if the limit is reached. If no ACK is received,
 165 the terminal must repeat the RACH access procedure.

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

174 A maximum number of repetitions for the RACH access
 175 procedure is defined, called k_{\max} . After k_{\max} attempts, a
 176 request is dropped. A repetition can be due to collision (with
 177 probability p_C) and to no ACK received (with probability
 178 $p_R(k)$ at the k -th attempt, with the associated power level).
 179 In addition, a maximum amount of time is defined for the
 180 completion of an access procedure instance. When this time
 181 is reached, a timeout expires, and the instance is dropped.

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

185 This whole procedure is illustrated in Fig. 1, where we see
 186 new service request generation on the left, the ACB subsystem,
 187 followed by the RACH, the Decoder and the Limiter, all
 188 with their backoffs, timeout possibilities and maximum num-
 189 ber of retries. The Network subsystem corresponds to service
 190 by the BS, if no blocking occurs. In the system, the following
 191 events lead to drop the connection attempt: network blocking,
 192 timeout, and exceeding the RACH retry limit.

193 B. Sliced System

194 In case of a sliced system, it is necessary to define an allo-
 195 cation of the BS resources to the different slices (identified in
 196 this article with a superscript denoting the slice index).

197 In the spirit of the 3GPP LTE standard, we assume that the
 198 barring probability is a characteristic of a service, but since

we allocate one service to each slice, the barring probabili- 199
 ty $p_A^{(i)}$ depends on the slice. The power ramping offset can 200
 provide a significant differentiation among slices, increasing 201
 the probability of decoding for the slices using higher power. 202
 For this reason we will consider different values for differ- 203
 ent slices. The subset of RACH preambles that can be used 204
 by a slice significantly impacts the collision probability. We 205
 will thus consider the case of different subsets (possibly with 206
 non-empty intersection) of preambles for different slices. We 207
 will instead assume that the ACKs provided by the BS to ser- 208
 vice requests that succeed on the RACH and at the decoder 209
 are equally available to all slices. Obviously, the maximum 210
 number of terminals in the RRC_CONNECTED state is a key 211
 aspect for governing the slice KPIs, and we will thus consider 212
 cases where values are different for different slices. These val- 213
 ues have an impact on the bandwidth share obtained by each 214
 terminal. Specifically, we assume that the BS allocates por- 215
 tions of bandwidth to each slice, and that the bandwidth is 216
 then equally shared within the slice among terminals in the 217
 RRC_CONNECTED state. Out of the M available positions, 218
 we reserve $M^{(i)}$ for unique use of slice i , with the sum of the 219
 $M^{(i)}$ less or equal to M . The remaining positions are shared 220
 by all slices. The values of backoff delays and access timeouts 221
 must be tailored to the types of service and the KPI goals of 222
 each slice, so they must be carefully set by the operator. 223

Note that we consider a partial isolation of slice resources, 224
 and we will show in the Numerical Results Section that this 225
 approach can provide better performance with respect to a full 226
 isolation. 227

228 III. ANALYSIS

229 We model a sliced system that represents the uplink chain
 230 that goes from the end user terminal, to the radio connection
 231 to the BS, to network service within a cell. We leave out of
 232 the analysis the connection from the BS to the core network
 233 and study in detail BS resource slicing.

234 A. System Flows

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

242 The ACB block sees a flow $\zeta^{(i)}$ as input. Barring at the
 243 ACB happens with probability $p_A^{(i)}$ and yields backoffs $A^{(i)}$
 244 for slice i , with no limit on the number of consecutive backoffs
 245 (up to the timeout values). We assume that, within a slice, ACB
 246 backoff durations are i.i.d. and exponentially distributed.

247 The RACH block receives the flow $\lambda^{(i)}$ from the ACB, which
 248 is no higher than $\zeta^{(i)}$ due to the possibility of timeout in the
 249 ACB. Failures on a RACH access attempt can be due to colli-
 250 sion, decoding errors or clipping at Limiter. A user cannot
 251 distinguish which type of failure occurred, it simply observes
 252 that the BS does not acknowledge its request in an interval
 253 T_{\max} and then it schedules a RACH backoff before another

attempt will start (if the timeout has not expired). We call *stage* k the k -th RACH access attempt. We assume that the RACH backoff durations $B^{(i)}$ are i.i.d. random variables (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 $\psi^{(i)} = \sum_{k=1}^{k_{\max}^{(i)}} \psi_k^{(i)}$.

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

The Limiter block 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 service 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 service request 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 $Y_j^{(i)}$, $j = 1, 2, \dots, k_{\max}^{(i)}$, $i = 1, 2, \dots, S$. 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 the 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. Instead, in case of timeout, the time spent is the timeout value selected for slice i , namely $T_O^{(i)}$, and the request is dropped as well. The distribution of $Y_k^{(i)}$ is

$$F_{Y_k^{(i)}}(x) = \Pr \left\{ \sum_{n=1}^{L^{(i)}} A_n^{(i)} + kT_{\max} + \sum_{n=1}^k B_n^{(i)} \leq x \right\}, \quad (1)$$

where $L^{(i)} \geq 0$ is the random number of back-to-back deferrals experienced because of ACB, due to the barring probability $p_A^{(i)}$ associated to slice i , and the subscript n indicates the n -th passage through either the backoff of ACB or RACH. Similarly, the distribution of $Y_{k-1}^{(i)} + Z$ is

$$F_{Y_{k-1}^{(i)}+Z}(x) = \Pr \left\{ \sum_{n=1}^{L^{(i)}} A_n^{(i)} + (k-1)T_{\max} + \sum_{n=1}^{k-1} B_n^{(i)} + Z \leq x \right\} \quad (2)$$

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

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

The above considerations tell that the distribution (1) can be obtained as the convolution of (4) with an Erlang distribution with shape parameter equal to k and average $k E[B^{(i)}]$ (i.e., the average of k backoff intervals), and a time shift $k T_{\max}$.

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:

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

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.

350 D. Event Probabilities

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

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

$$357 \quad P_S^{(i)}(k) = P_N^{(i)}(k) \left(1 - p_C^{(i)}\right) \left(1 - p_R^{(i)}(k)\right) \left(1 - p_B^{(i)}\right) \\ 358 \quad \times F_{Y_{k-1}^{(i)}+Z} \left(T_O^{(i)}\right). \quad (6)$$

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

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

$$363 \quad P_B^{(i)}(k) = P_N^{(i)}(k) \left(1 - p_C^{(i)}\right) \left(1 - p_R^{(i)}(k)\right) p_B^{(i)} \\ 364 \quad \times F_{Y_{k-1}^{(i)}+Z} \left(T_O^{(i)}\right). \quad (7)$$

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

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

$$372 \quad P_{TO}^{(i)}(k) = P_N^{(i)}(k) \left\{ \left(1 - p_C^{(i)}\right) \left(1 - p_R^{(i)}(k)\right) \right. \\ 373 \quad \times \left[1 - F_{Y_{k-1}^{(i)}+Z} \left(T_O^{(i)}\right) \right] \\ 374 \quad + \left[1 - \left(1 - p_C^{(i)}\right) \left(1 - p_R^{(i)}(k)\right) \right] \\ 375 \quad \times \left[1 - F_{Y_k^{(i)}} \left(T_O^{(i)}\right) \right] \left. \right\}. \quad (8)$$

376 The total timeout probability observed by a slice is therefore

$$377 \quad P_{TO}^{(i)} = \sum_{k=0}^{k_{\max}^{(i)}} P_{TO}^{(i)}(k). \quad (9)$$

378 *Busy tone*: Access requests that exceed the RACH retry
379 limit, experience a network blocking event, or a timeout, are
380 dropped. Therefore, the busy tone is sent with probability $1 -$
381 $P_S^{(i)} = P_N^{(i)}(k_{\max}^{(i)} + 1) + P_B^{(i)} + P_{TO}^{(i)}$.

382 E. Derivation of Throughputs and Loads With Cycles

383 With the expressions derived so far, we have characterized
384 the trajectory of the exogenous access requests that feed the
385 system for slice i , i.e., $\zeta^{(i)}$. However, the expressions derived
386 are functions of three parameters that we need to derive next:
387 $p_C^{(i)}$, $p_R^{(i)}(k)$, and $p_B^{(i)}$.

388 *RACH collision probability and throughput*: The input of
389 RACH is the flow $\lambda^{(i)}$ that arrives from ACB. However, RACH
390 has internal cycles, and $\lambda^{(i)}$ is just the input to the first stage.

With the definitions of Section III-C, we have the following
input flows for each successive stage (note that $\lambda_1^{(i)} = \lambda^{(i)}$):

$$393 \quad \lambda_k^{(i)} = \zeta^{(i)} P_N^{(i)}(k), \quad k = 1, 2, \dots, k_{\max}^{(i)}. \quad (10)$$

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

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

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

$$405 \quad \ell^{(i)} = \frac{\zeta^{(i)}}{N^{(i)} + N_s} \sum_{k=1}^{k_{\max}^{(i)}} P_N^{(i)}(k). \quad (11)$$

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

$$408 \quad p_{C,j} = \begin{cases} 1 - e^{-\tau \ell^{(i)}}, & 1 \leq i \leq S, \text{ dedicated preamble;} \\ 1 - e^{-\tau \sum_{i=1}^S \ell^{(i)}}, & \text{shared preamble.} \end{cases} \quad (12) \quad 409$$

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

$$413 \quad p_C^{(i)} = 1 - \frac{N^{(i)} e^{-\tau \ell^{(i)}} + N_s e^{-\tau \sum_{q=1}^S \ell^{(q)}}}{N^{(i)} + N_s}. \quad (13)$$

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

$$415 \quad \psi_k^{(i)} = \left(1 - p_C^{(i)}\right) \lambda_k^{(i)}, \quad \psi^{(i)} = \sum_{k=1}^{k_{\max}^{(i)}} \psi_k^{(i)}. \quad (14)$$

416 *Throughput of Decoder*: At each stage of the RACH,
417 Decoder has a different failure probability, due to power
418 ramping in RACH message transmissions [8]. In particular, as
419 explained in [8], the Decoder failure probability is expressed
420 as a negative exponential of the power level index (an integer
421 starting from 1 and incremented at each RACH access) used
422 for the transmission. This is the negative exponential of the
423 number of attempts under standard operations, while in our
424 case the expression becomes $e^{-k-o^{(i)}}$, where k is the RACH
425 attempt stage and $o^{(i)} \geq 0$ is an integer representing the slice
426 offset, i.e., the number of steps in the power ramping proce-
427 dure which are skipped by slice i . Therefore, at stage k , slice
428 i observes the following Decoder throughput:

$$429 \quad \phi_k^{(i)} = \psi_k^{(i)} \left(1 - e^{-(k+o^{(i)})}\right) \quad (15)$$

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

431 *Losses due to Limiter*: The BS can only grant Θ requests
432 per RAO, shared between the slices. Therefore there are losses
433 when the output of Decoder in a RAO interval is higher
434 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 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 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). Thus, for each preamble, the average output per RAO, after decoding, is

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

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

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

For a dedicated pool we have the following distribution:

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

while for the shared pool of preambles we have

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

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

$$\Omega_a = \sum_{a_1=0}^{N_p} \sum_{a_2=0}^{N_p} \dots \sum_{a_S=0}^{N_p} \omega_{a_1}^{(1)} \omega_{a_2}^{(2)} \dots \omega_{a_S}^{(S)} \omega_{s(a-\sum_{r=1}^S a_r)}. \quad (20)$$

Overall, the average number of losses is

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

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

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

The resulting per-slice `Limiter` throughput is

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

Since losses at `Limiter` do not discriminate between RACH stages, the `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 :

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

F. A Model for the Network Subsystem in a Sliced BS

The `Network` subsystem is a BS network processor. It can serve at most M users at the same time, and the M available positions in service must be shared among slices. Each slice is granted exclusive access to $M^{(i)}$, $i = 1, 2, \dots, S$ positions, with $\sum_{i=1}^S M^{(i)} = M - M_s$, and $M_s \geq 0$. If not all positions are dedicated (i.e., $M_s > 0$), remaining positions are shared among all slices. Arrivals that do not find available service positions are dropped, thus originating the network blocking probability.

The `Network` subsystem has total service capacity C services per second, out of which $C^{(i)}$, $i = 1, 2, \dots, S$, is reserved for slice i , and C_s is shared among slices. If `Network` is serving up to $M^{(i)}$ users for slice i , they equally share $C^{(i)}$. However, when there are $m^{(i)} > M^{(i)}$ customers, the i -th queue obtains a service rate equal to $C^{(i)}$ plus a portion of the shared capacity C_s proportional to $m^{(i)} - M^{(i)}$. Hence, the i -th queue service rate depends on the total number of services of all slices, i.e., the service rate when $m^{(i)} > M^{(i)}$ is $C_i + C_s / \sum_{i=1}^S \max(0, m^{(i)} - M^{(i)})$. This service policy is reasonable when most of resources are dedicated to slices, and the fraction of shared capacity is small. Other policies can be considered in different scenarios, and studied with approaches similar to what we describe below.

The above service policy description resembles the operations of a multi-class processor sharing (PS) queue in which a class receives part of what cannot be used by other classes. Thus, we model the network processor of the BS with a PS queue with S classes of customers 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 the shared resources are accessed only when all dedicated positions are busy.

Since the resulting queueing system, where the service rate of one class depends not only on the number of customers in that class, but also in other classes, does not admit a product-form solution (PFS) [9], we study this queue by means a continuous-time Markov chain with S -dimensional state space (one dimension per class, to count the number of services in the class) whose transitions are depicted in Fig. 2 for the case of two slices ($S = 2$), hence two customer classes. In the figure, each state of the chain reports the number of customers

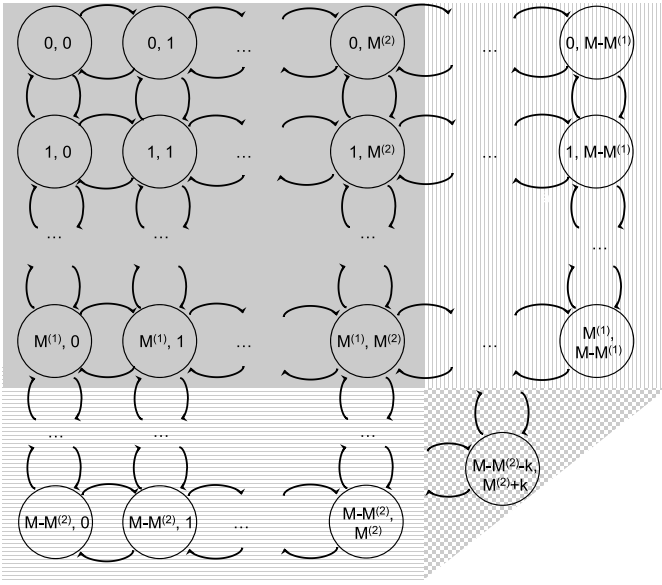


Fig. 2. CTMC describing the operation of Network for $S = 2$. Top-to-bottom transitions have rate $\sigma^{(1)}$, while left-to-right transitions have rate $\sigma^{(2)}$. Bottom-to-top transitions have rate $C^{(1)}$ in the first $M^{(1)}$ rows and $C^{(1)} + Q^{(1)}(a, b)$ in the remaining rows. Right-to-left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

524 $m^{(1)} \leq M - M^{(2)}$ in slice 1 and $m^{(2)} \leq M - M^{(1)}$ in slice
 525 2, subject to the constraint that $m^{(1)} + m^{(2)} \leq M$. The chain
 526 has a precise symmetry and a pentagonal shape, which is due
 527 to the above constraints.

528 If we denote by (a, b) the chain's state, where a is the
 529 number of services in slice 1 and b is the number in slice 2,
 530 the transition rates in Fig. 2 from state (a, b) to other states
 531 are as follows:

$$\begin{cases}
 (a+1, b): \sigma^{(1)} \quad \forall a \leq M^{(1)} - 1, b \leq M^{(2)} \\
 \quad \text{and } \forall a \leq M - M^{(2)} - b - 1 \\
 (a, b+1): \sigma^{(2)} \quad \forall b \leq M^{(2)} - 1, a \leq M^{(1)} \\
 \quad \text{and } \forall b \leq M - M^{(1)} - a - 1 \\
 (a-1, b): C^{(1)} \quad \forall a \leq M^{(1)} \\
 (a-1, b): C^{(1)} + Q^{(1)}(a, b) \quad \forall M^{(1)} < a \leq M - M^{(2)} \\
 (a, b-1): C^{(2)} \quad \forall b \leq M^{(2)} \\
 (a, b-1): C^{(2)} + Q^{(2)}(a, b) \quad \forall M^{(2)} < b \leq M - M^{(1)}
 \end{cases}
 \quad (25)$$

534 For simplicity of notation, we used the following quantities:

$$\begin{aligned}
 535 \quad Q^{(1)}(a, b) &= \begin{cases} \frac{a - M^{(1)}}{a - M^{(1)} + \max(0, b - M^{(2)})} & \text{if } a > M^{(1)}; \\ 0 & \text{otherwise;} \end{cases} \quad (26) \\
 536 \quad Q^{(2)}(a, b) &= \begin{cases} \frac{b - M^{(2)}}{b - M^{(2)} + \max(0, a - M^{(1)})} & \text{if } b > M^{(2)}; \\ 0 & \text{otherwise.} \end{cases} \quad (27)
 \end{aligned}$$

537 Note that different service policies (i.e., different approaches
 538 to access the shared capacity) can be accommodated in this
 539 Markov chain by just modifying transition rates.

540 The solution of the Markov chain can be obtained numerically
 541 with specialized tools like SMART [10]. We therefore
 542 solve the Markov chain numerically to compute $p_B^{(i)}$ as the
 543 sum of the relevant state probabilities. In the case of Fig. 2,

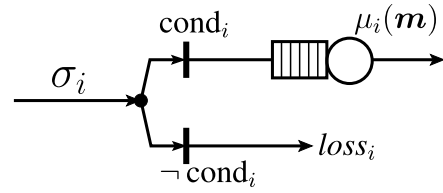


Fig. 3. Representation of the i -th queue: arrivals from Limiter are queued if condition (29) is satisfied, and dropped otherwise.

for slice 1 we sum over states that lay on the bottom and
 diagonal edges, whereas for slice 2 we sum over states on the
 edges on the right and on the diagonal. This is because the
 diagonal edge at the right-bottom part of the chain in Fig. 2
 contains states in which both slices suffer blocking.

The state space size of the Markov chain in Fig. 2,
 is $O(M^S)$, where M is the maximum number of
 RRC_CONNECTED terminals and S is the number of slices.
 Hence, being M of the order of few hundreds in real systems,
 the numerical solution of the Markov chain is a viable
 approach only for a (very) small number of slices.

G. Bounds on the Performance of the Network Subsystem

To mitigate the computational complexity problem, and thus
 be able to handle a higher number of slices, we can observe
 that the queuing model has, for each customer class, one
 queue with finite capacity constraints, Poisson arrivals, and
 state dependent processor service discipline. Recalling that the
 arrival rate at the i -th queue is equal to $\sigma^{(i)}$ and the available
 buffer size is equal to $M - \sum_{j=1, j \neq i}^S M^{(j)}$, and denoting by
 $\mathbf{m} = [m^{(1)}, m^{(2)}, \dots, m^{(S)}]$ the S -dimensional vector of the
 current states of queues, and by $\mu_i(\mathbf{m})$ the state dependent
 service rate of the i -th processor sharing queue, we have that

$$\mu_i(\mathbf{m}) = \begin{cases} C^{(i)} & \text{if } m^{(i)} \leq M^{(i)} \\ C^{(i)} + C_s \frac{m^{(i)} - M^{(i)}}{\sum_{j=1}^S \max\{0, m^{(j)} - M^{(j)}\}} & \text{otherwise.} \end{cases} \quad (28)$$

The previous expression for the service rate accounts for the
 two cases; (i) for $m^{(i)} \leq M^{(i)}$ the reserved capacity $C^{(i)}$
 of the i -th slice is equally shared among the $m^{(i)}$ customers;
 (ii) when there are $m^{(i)} > M^{(i)}$ customers, the i -th queue obtains
 a service rate equal to $C^{(i)}$ plus a portion of the shared capacity
 C_s proportional to $m^{(i)} - M^{(i)}$. Moreover, since Network
 can serve at most M customers, for all feasible states $\mathbf{m} =$
 $[m^{(1)}, m^{(2)}, \dots, m^{(S)}]$ of the queueing system, we have that
 $\sum_{i=1}^S m^{(i)} \leq M$ and $\sum_{j=1}^S \max\{0, m^{(j)} - M^{(j)}\} \leq M_s$.

Arriving customers that cannot enter the i -th queue because
 of lack of space in the reserved positions in the queue (i.e.,
 because they find $m^{(i)} = M^{(i)}$), or because all the shared
 positions are full (i.e., because they find $\sum_{j=1}^S \max\{0, m^{(j)} -$
 $M^{(j)}\} = M_s$) are lost.

Fig. 3 depicts the i -th queue (for $i = 1, \dots, S$) of a queueing
 system with these features. The two vertical bars labeled as
 'cond _{i} ' and '¬cond _{i} ' represent the two situations that an

arriving customer can encounter. In particular,

$$\text{cond}_i = \left(m^{(i)} < M^{(i)} \right) \text{ or } \left(\sum_{j=1}^S \max\{0, m^{(j)} - M^{(j)}\} < M_s \right). \quad (29)$$

If the first part of the condition does not hold, the arriving customer might be served by using the shared resources (M_s and C_s), but if the whole condition cond_i does not hold (case ‘ $\neg \text{cond}_i$ ’), the arriving customer is lost, as pictorially illustrated in Fig. 3.

Several features of this model make it a non-standard multi-class queuing system (i.e., a queuing system which does not admit a PFS), where the main non-product-form characteristic concerns the rate dependency in (28). In particular, the rate of the i -th queue (for $i = 1, \dots, S$) depends on the number of customers in all the S queues. From this follows that the derivation of the blocking probabilities requires the solution of the underlying continuous-time Markov chain (CTMC), but, as we already noted, this is feasible only for very small S (in [1] we solved the model for $S = 2$).

To avoid the complexity of a solution based on the generation of the entire state space of the Network model, we propose a simple modification of the model that allows to come up with a PFS queuing network. With this transformation we will be able to derive a lower and an upper bound on the blocking probabilities.

To derive the PFS bounds we modify the rate $\mu_i(\mathbf{m})$ of the processor sharing server for each slice $i = 1, \dots, S$ defined in (28) by removing the dependency on the number of customers in all the other $S - 1$ queues. In other words, we replace $\mu_i(\mathbf{m})$ with $\mu_i^{(l)}(m^{(i)})$ (resp. $\mu_i^{(u)}(m^{(i)})$) to derive a lower bound (resp. an upper bound) on the loss probability. We derive a lower bound on the loss probability by assuming that when $m^{(i)} > M^{(i)}$ customers are in the queue, $M^{(i)}$ of them get a service rate $\frac{C^{(i)}}{M^{(i)}}$ and the remaining $m^{(i)} - M^{(i)}$ customers access shared resources with no competition from the users of other slices. Thus, we assume that Network allocates all shared bandwidth resources to slice i , i.e., that slice i is the only one accessing the shared bandwidth:

$$\mu_i^{(l)}(m^{(i)}) = \begin{cases} C^{(i)} & \text{if } m^{(i)} \leq M^{(i)} \\ C^{(i)} + C_s & \text{otherwise.} \end{cases} \quad (30)$$

To derive an upper bound for loss probabilities we focus our attention on the second case of (28). In particular, we can see that $M - \sum_{j=1}^S M^{(j)} \geq \sum_{j=1}^S \max\{0, m^{(j)} - M^{(j)}\}$, and hence we can write that

$$\mu_i^{(u)}(m^{(i)}) = \begin{cases} C^{(i)} & \text{if } m^{(i)} \leq M^{(i)} \\ C^{(i)} + C_s \frac{m^{(i)} - M^{(i)}}{M_s} & \text{otherwise.} \end{cases} \quad (31)$$

In this case we assume that all slices are using the shared bandwidth resources. Note that the two bounds are reached for some of the possible states of the queuing system.

By using (30), and (31) instead of (28), we can then derive lower and upper bounds for the loss probabilities (that generate, respectively, upper and lower bounds on throughput). Remarkably, the computation of the bounds can be done by

exploiting their PFS. Indeed, to compute the measures we are interested in, we can use the efficient normalization constant computation algorithm proposed in [11]² whose computational complexity is $O(S \cdot M^2)$, where M is the maximum number of users in the RRC_CONNECTED state and S is the number of slices.

Note that the bounds are not directly applicable to different service disciplines (i.e., different algorithms for accessing the shared BS capacity), but approaches similar to the one just described can be devised also for other cases.

The main question of any bounding technique concerns the tightness of the provided bounds. To answer this question we will show various numerical examples in the performance evaluation section. Here we remark the fact that the bounds are different only if the slices can use shared resources, and the distance between lower and upper bounds depends on the quantity of shared capacity and positions in the Network subsystem. However, for all those cases in which the RACH subsystem constitutes the main performance bottleneck, shared resources at Network remain unused, so that the bounds (practically) coincide and yield the exact solution.

H. 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:

$$F_{T^{(i)}}(x) = P_{TO}^{(i)} \mathcal{U}(x - T_O^{(i)}) + \sum_{k=1}^{k_{\max}^{(i)}} \frac{F_{Y_{k-1}^{(i)}+Z}(x)}{F_{Y_{k-1}^{(i)}+Z}(T_O^{(i)})} \times \left(P_S^{(i)}(k) + P_B^{(i)}(k) \right) + P_N^{(i)} \left(k_{\max}^{(i)} + 1 \right) \times \frac{F_{Y_{k_{\max}-1}^{(i)}}(x - T_{\max})}{F_{Y_{k_{\max}-1}^{(i)}}(T_O^{(i)} - T_{\max})}, \quad (32)$$

where $\mathcal{U}(x - T_O^{(i)})$ is a unit step at time $T_O^{(i)}$. As a corollary of (32), note that the CDF of the time spent for one service request in case of success is

$$F_{T^{(i)}}(x | \text{Success}) = \frac{1}{P_S^{(i)}} \sum_{k=1}^{k_{\max}^{(i)}} \frac{F_{Y_{k-1}^{(i)}+Z}(x)}{F_{Y_{k-1}^{(i)}+Z}(T_O^{(i)})} P_S^{(i)}(k). \quad (33)$$

²That paper presents a computational efficient algorithm for a class of product-form models that represent a generalization of the PFS queuing networks with population size constraints.

TABLE II
SLICE PARAMETERS FOR THE CONSIDERED APPLICATION SCENARIOS WITH TWO SLICES (COLUMNS LABELED AS ‘H’ REFER TO HTC DEDICATED RESOURCES, ‘M’ IS USED FOR MTC AND ‘S’ FOR SHARED RESOURCES)

Scenario	Access request share		Population share		Capacity [Mb/s]			Preambles			Positions			Timeout [s]	
	H	M	H	M	H	M	S	H	M	S	H	M	S	H	M
Sparse IoT	0.95	0.05	0.79	0.21	80	2	18	40	5	9	100	10	90	5	5
Dense IoT	0.05	0.95	0.05	0.95	75	5	20	10	40	4	40	100	60	3	1
Small factory	0.75	0.25	0.97	0.03	50	20	30	30	10	14	50	50	100	5	0.1
Big factory	0.3	0.7	0.81	0.19	10	50	40	10	30	14	20	150	30	5	0.1

IV. 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 messages of HTC type with average length 1.2 Mb, and with average length 8 kb in case of MTC; 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 four instances of a BS with two slices, one carrying MTC traffic and one HTC, the shares of traffic (referred to the intensity of access requests, population (referred to the number of devices) and BS capacity dedicated to the two slices. The table also reports the number of dedicated RACH preambles and positions in the RRC_CONNECTED state, and the timeout values. The resources which are not dedicated to slices can be evenly shared. Later we will also address the case of coexistence of larger numbers of slices, considering examples with three and six slices, respectively. The two-slice scenarios that we consider are as follows:

- **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 access requests characterized by low traffic, so that a large part of the capacity is used by a slice with few HTC devices.
- **Small Factory** – A BS serving a 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.

In the numerical evaluation, we assume that HTC devices on average, considering video, voice and data applications, generate one message per second, i.e., each device offers a traffic of 1.2 Mb/s, while for MTC devices the timeout corresponds to the message generation interval, so to emulate the process of updating the status of a system. This corresponds to generating 8 to 80 kb/s per MTC device in the settings of Table II, but we will also explore more extreme cases, generating 0.8 to 800 kb/s per MTC device. Table II reports, for each slice and each scenario, both the share of access requests and the corresponding share of population of devices, computed according to the message generation rate of each HTC or MTC device.

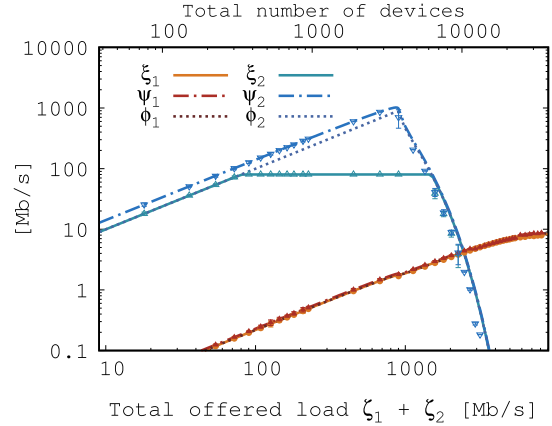


Fig. 4. Small Factory throughput (in Mb/s) at RACH (Ψ), Decoder (ϕ) and Network (ξ) vs. offered load (in Mb/s and in number of devices); lines represent the model, markers with confidence intervals are for simulations; HTC in blue, MTC in red.

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

Fig. 4 compares analytical and simulation results, with their 95% confidence intervals. Notwithstanding the approximations introduced in the model, the figure shows a very good match between model and simulation, up to very high offered loads (more than 7 Gb/s as shown in the x-axis of the figure), which corresponds to large device populations (up to a few tens of thousands of devices, as shown in the secondary x-axis, which maps the offered load onto the number of devices).

B. Throughput

Figs. 4 to 7 illustrate the behavior of the system throughput (at RACH, Decoder and Network) for the four considered two-slice scenarios. In the Sparse IoT case, HTC saturates first, and the HTC load on 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

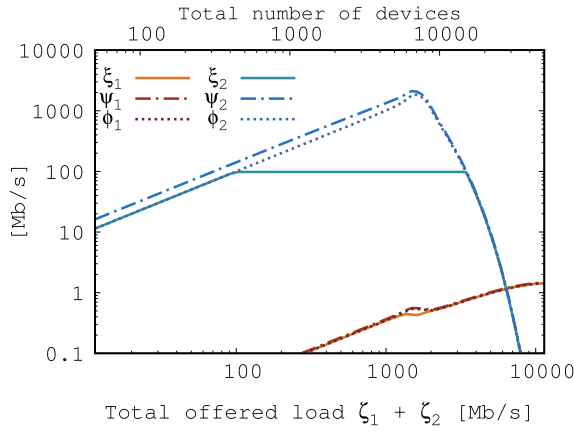


Fig. 5. Sparse IoT throughput (in Mb/s) at RACH (Ψ), Decoder (ϕ) and Network (ξ) vs. offered load (in Mb/s and in number of devices); HTC in blue, MTC in red.

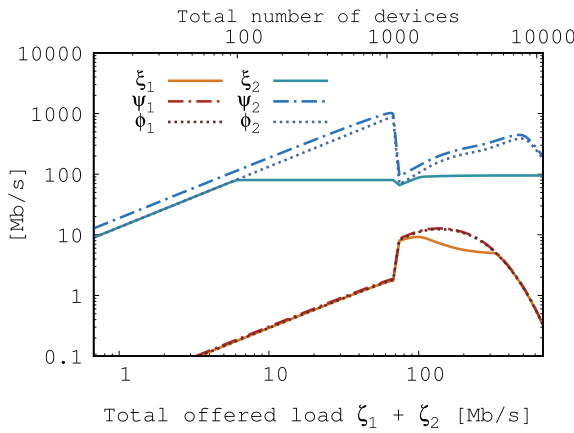


Fig. 6. Dense IoT throughput (in Mb/s) at RACH (Ψ), Decoder (ϕ) and Network (ξ) vs. offered load (in Mb/s and in number of devices); HTC in blue, MTC in red.

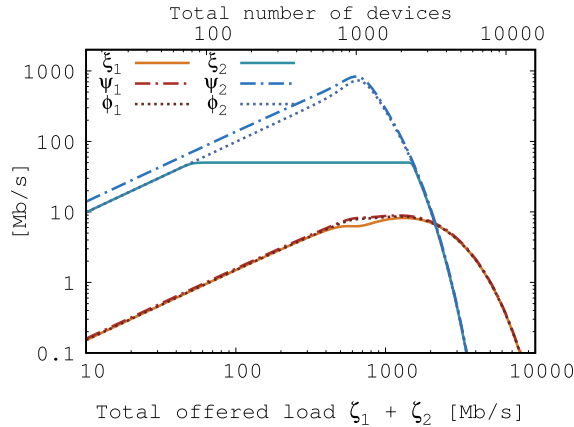


Fig. 7. Big factory throughput (in Mb/s) at RACH (Ψ), Decoder (ϕ) and Network (ξ) vs. offered load (in Mb/s and in number of devices); HTC in blue, MTC in red.

748 the MTC and HTC saturation regions superpose. Here, the
 749 activity of MTC in RACH heavily affects HTC performance.
 750 Therefore, supporting the coexistence of HTC and MTC slices
 751 in such scenarios is challenging. If we go back to the Small
 752 Factory case used above for validation, we notice only minor

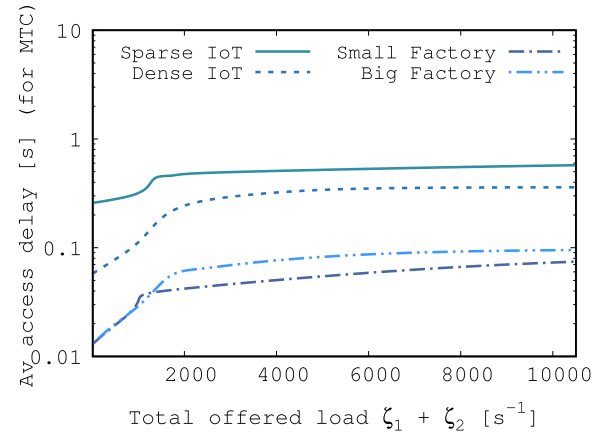


Fig. 8. Average access delays for slice 1 (MTC) in seconds for the four different scenarios versus the total load offered to the BS in service requests per second.

753 differences with the Sparse IoT case. However, in this case,
 754 the limited resources allocated to HTC make it easier to avoid
 755 impairments for MTC. More critical is the Big Factory case, in
 756 which the MTC traffic is predominant and yet a small amount
 757 of HTC connections can seriously hinder MTC performance at
 758 relatively low aggregate traffic rates, while under heavy traffic
 759 the impact of HTC on the throughput of MTC is less relevant.

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

766 Due to the heterogeneity of slices, there are no clear optimal
 767 device population sizes at which all slices receive the maxi-
 768 mal throughput. In all cases, the HTC slice works better with
 769 a few hundreds of devices, while MTC achieves the high-
 770 est Network throughput with a few thousands of devices.
 771 Considering the Dense IoT scenario, which shows the highest
 772 density of MTC devices (95% of the total), our study shows
 773 that a cell can sustain about 2000 MTC devices. If we con-
 774 sider instead the Small Factory case, with the highest
 775 density of HTC devices (97%), we can observe that the Network
 776 subsystem of a cell saturates with as few as 300 HTC devices.
 777

C. Access Delay

778 The numerical results of access delay (the time elapsing
 779 from the request generation until the start of the message trans-
 780 mission) for MTC and HTC traffic are shown in Figs. 8 and 9,
 781 respectively. Note that in this case we use service requests per
 782 second as the horizontal axis metric; this is necessary because
 783 the throughput per access request is different in the four two-
 784 slice scenarios that we compare: in Sparse IoT each access
 785 request corresponds on average to 1.14 Mb, in Dense IoT to
 786 67.6 kb, in Small Factory to 902 kb, and in Big Factory to
 787 365.6 kb. In the case of MTC, the two curves for Small Factory
 788 and Big Factory saturate at 100 ms, which is the timeout for
 789 those cases. The other two cases remain well below their time-
 790 out values which are much less stringent. In the case of HTC,
 791

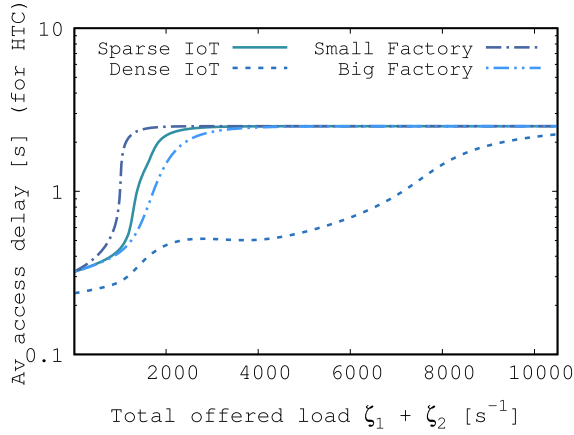


Fig. 9. Average access delays for slice 2 (HTC) in seconds for the four different scenarios versus the total load offered to the BS in service requests per second.

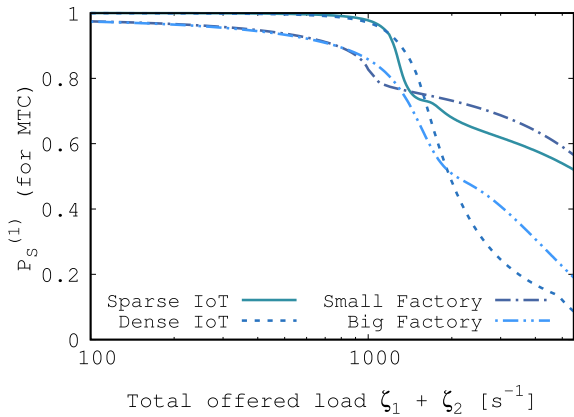


Fig. 10. Success probability for MTC for the four different scenarios versus the total load offered to the BS in service requests per second.

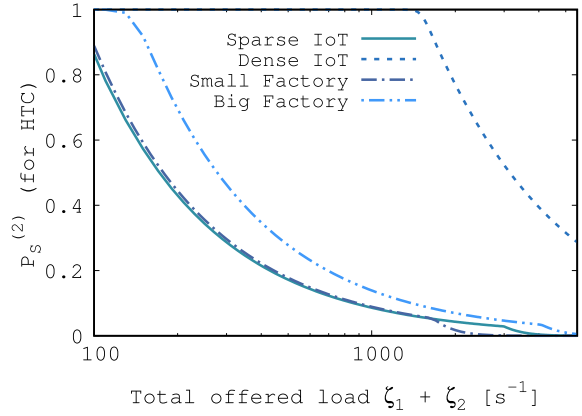


Fig. 11. Success probability for HTC for the four different scenarios versus the total load offered to the BS in service requests per second.

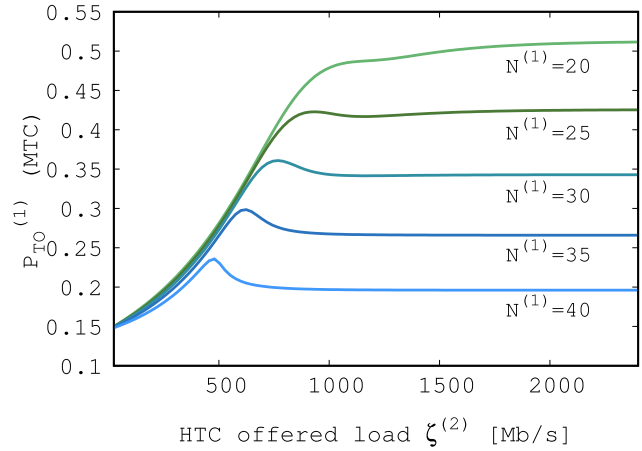


Fig. 12. Timeout probability for MTC at fixed MTC service request rate $\zeta^{(1)} = 8$ Mb/s versus the load offered by HTC in Mb/s.

791 we see that all curves saturate at the same value, which is
 792 close to 2.25 s, due to the maximum permitted number of
 793 retries, and the average backoff delay equal to 0.25 s. To this
 794 we must add $10 \cdot T_{\max}$, which is however just about 0.13 s.
 795 The Small Factory scenario saturates first because a large frac-
 796 tion of the BS traffic is associated with only a small portion of
 797 dedicated resources. The Dense IoT scenario yields the lowest
 798 delays because its traffic share is very low, and the reserved
 799 resources prove to be sufficient to achieve low delay.

800 *D. Success Probability*

801 Success probabilities for the four considered scenarios are
 802 presented in Figs. 10 and 11 for MTC and HTC traffic, respec-
 803 tively. In the MTC case, Small Factory and Big Factory suffer
 804 from the very low timeout values, but achieve good success
 805 probabilities up to about 1000 requests/s. Beyond this value,
 806 the RACH subsystem approaches saturation, and retrials make
 807 timeouts more likely. In the case of HTC, we see very high
 808 success rates in the Dense IoT scenario, due to the fact that
 809 the HTC traffic share in this case is very low, and resources
 810 reserved to HTC are largely sufficient.

811 *E. Lesson Learnt From the Two-Slice Scenarios*

812 One of our key observations is that the saturation of RACH
 813 is a critical issue, and unexpected behaviors are observed for
 814 the traffic loads that bring the RACH to saturation. In Fig. 12
 815 we plot the probability of reaching the timeout for MTC traf-
 816 fic versus the HTC traffic load in the Big Factory scenario,
 817 assuming that the MTC traffic is fixed at 8 Mb/s, and that
 818 the number of preambles reserved for MTC is varied between
 819 20 and 40. We clearly see a bump in the timeout probab-
 820 ility that corresponds to HTC traffic values that lead to RACH
 821 saturation. After this point, HTC consumes little Network
 822 resources, but saturates the RACH, so that there is a clear need
 823 to protect MTC by allocating a large number of dedicated
 824 preambles. If the number of dedicated preambles is too small,
 825 the timeout probability settles at unacceptable values.

826 Moreover, considering throughput and delay figures, with
 827 the parameters used in the discussed experiments which are
 828 typical of networks slowly evolving towards 5G, we can expect
 829 to sustain populations of at most a few hundreds of HTC
 830 devices or a few thousands of MTC devices. Of course, as we
 831 can see from Figs. 10 and 11, approaching such limit num-
 832 bers implies lower success probabilities. This gives rise to a
 833 tradeoff for the network operator, that can exploit accurate

TABLE III
CONFIGURATION FOR THE CASE OF THREE SLICES WITH HIGHLY LOADED HTC SLICES AND LIGHT MTC TRAFFIC

Slice	Access request share	Population share	Capacity [Mb/s]	Preambles	Positions	Timeout [s]	Duty Cycle [s]
HTC (operator 1)	0.6	0.66	40	30	40	5	1
HTC (operator 2)	0.3	0.33	20	15	20	5	1
MTC (IoT)	0.1	0.01	10	5	5	0.1	0.1
Shared	N/A	N/A	30	4	135	N/A	N/A

834 modeling techniques, like the one we propose in this article,
835 to select the operational point for the network.

836 F. Increasing the Number of Slices

837 We now analyze more complex scenarios, involving the
838 coexistence of more than two slices. Since the complexity of
839 the corresponding CTMC describing the Network subsystem scales exponentially, we use these examples to validate
840 the alternative approach proposed in Section III-G, which is
841 based on the use of smart, non-trivial bounds. We performed
842 an extensive set of numerical experiments to compare the
843 performance measures obtained with the bounding technique
844 against those that can be derived by using (when feasible) the
845 CTMC approach.
846

847 1) *A First Three-Slice Scenario:* We first consider a sce-
848 nario with light MTC traffic and two highly loaded HTC slices.
849 The scenario could represent a cell with two tenants for public
850 data access plus an IoT operator, with the realistic parameter
851 set shown in Table III.

852 Note that, since it can be expected that the bounds are
853 extremely tight when the fraction of shared resources is
854 small, we leave unassigned 30% of capacity and two thirds
855 of Network positions. Like for previous experiments with
856 two slices, the BS capacity is set to 100 Mb/s, HTC ser-
857 vice requests correspond to 1.2 Mb messages on average, and
858 MTC traffic corresponds to small messages of 8000 bits. In
859 this case, the offered load fractions for the three slices are
860 0.6 and 0.3 for HTC slices, and 0.1 for the MTC slice (this
861 corresponds to having 99% of devices on the HTC slices).
862 The reserved capacities for the three slices are respectively
863 40, 20 and 10 Mb/s, so that 30 Mb/s are shared. Note that
864 the share of traffic offered by MTC is less than the share of
865 capacity reserved to MTC at the Network, so MTC is not
866 expected to use much of shared capacity unless all slices are
867 overloaded. Out of the 54 available RACH preambles, 30 plus
868 15 are allocated to HTC slices, and 5 to the MTC slice. Of
869 the 200 positions in the RRC_CONNECTED state, 40 and 20
870 respectively are allocated to the two HTC slices, and 5 to the
871 MTC slice, so that 135 positions are shared. The timeouts for
872 the access requests of the two HTC slices are equal to 5 s,
873 while the timeout for the MTC slice is 0.1 s.

874 The curves in Fig. 13 plot the throughput in terms of exact
875 values and bounds, expressed in Mb/s of the three slices as
876 a function of the total offered load and, in the secondary x-
877 axis, the total number of devices. The two HTC slices saturate
878 their available capacity, as shown by the flat part of the curves,
879 while the MTC slice, because of its much smaller message size

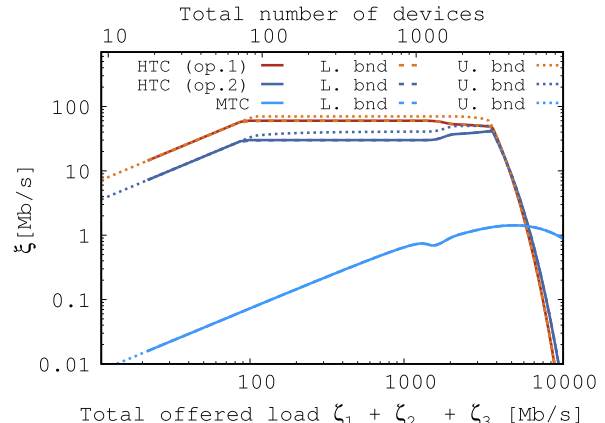


Fig. 13. Exact Network throughput values and bounds for the three-slice scenario of Table III vs. the total load (in Mb/s). The secondary x-axis reports the total number of devices resulting from the composition of the three slices.

and its dedicated resources, does not reach saturation. Since
we have a net predominance of HTC devices, the population
that can be efficiently sustained includes at most about one
hundred devices.

The bounds are indistinguishable from the exact solution in the case of the MTC slice, as well as for the two HTC slices before saturation. The bounds can be seen to be tight also for the two HTC slices when in saturation, where the exact solution almost overlaps with the lower bound. This is because the two slices saturate at the same offered load value, so that the condition in (31) is satisfied with high probability. The upper bound is not far either, at least in this case in which the MTC does not practically use shared resources, while the two HTC slices contend for shared resources. Fig. 14 shows that blocking probability bounds are very tight for the HTC slice with higher throughput, while they are slacker for the other HTC slice. Note that the lower bound for throughput is an upper bound for the blocking probability and vice versa (for consistency, the terms ‘lower’ and ‘upper’ used in the labels of the figures refer to throughput bounds). The only region in which bounds can be large is a small interval around the first knee of the throughput curve, where the blocking probability is however low.

2) *A Second Three-Slice Scenario:* We now consider a different three-slice scenario, one in which HTC and MTC traffic are comparable, no slice receives enough dedicated resources to serve all its traffic as it grows, and in which shared resources are abundant (i.e., 95% of capacity and 50% of positions are shared). Specifically, we consider one slice for HTC and two

TABLE IV
CONFIGURATION FOR THE CASE OF THREE SLICES WITH HIGH MTC TRAFFIC AND ABUNDANT DEDICATED RESOURCES

Slice	Access request share	Population share	Capacity [Mb/s]	Preambles	Positions	Timeout [s]	Duty Cycle [s]
HTC (public)	0.5	0.90	from 2.5 to 49.5	25	50	5	1
MTC (operator 1)	0.25	0.045	from 1.5 to 29.7	10	30	0.1	0.1
MTC (operator 2)	0.25	0.045	from 1 to 19.8	5	20	0.1	0.1
Shared	N/A	N/A	from 95 to 1	14	100	N/A	N/A

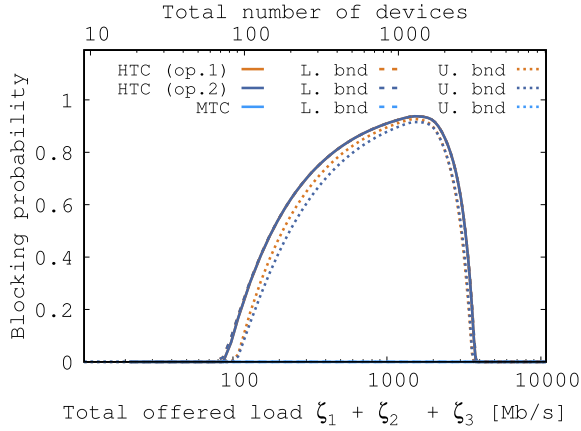


Fig. 14. Exact values and bounds for blocking probability in the three-slice scenario of Table III (MTC slices incur no blocking) vs. the total load (in Mb/s). The secondary x-axis shows the total number of devices resulting from the composition of the three slices.

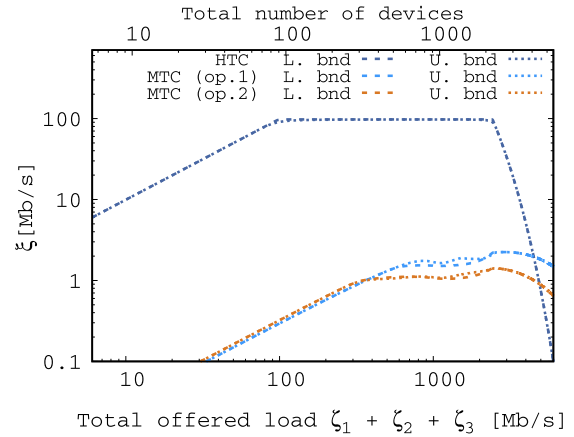


Fig. 15. Network throughput bounds for the three-slice scenario of Table IV with 2.5, 1.5, and 1 Mb/s of capacity assigned to the three slices (i.e., 95 Mb/s of shared capacity) versus the total load in Mb/s (lower x-axis), and total number of users (upper x-axis).

909 for MTC, with the parameters described in Table IV. Here the
 910 number of MTC devices is not negligible, as it accounts for
 911 about 9% of the population. The scenario could represent the
 912 case of an industrial site with a public data service on HTC and
 913 two MTC slices for two separate factories. We assume half of
 914 access requests are for HTC and the other half is equally split
 915 across the two MTC slices, although one MTC slice receives
 916 more resources than the other. Most of the capacity is left
 917 in the shared pool, because we have assumed that the HTC
 918 slice does not need stringent guarantees and the MTC devices
 919 need little capacity. The scenario was deliberately chosen with
 920 a large amount of shared capacity, so as to stress the differ-
 921 ences between bounds in the computation of the throughput
 922 of Network (see the impact of C_s on the bounds (30)–(31)).

923 Notwithstanding the large amount of shared resources,
 924 Fig. 15 shows that, for this case, the two bounds are very
 925 close. Limited differences can be noted for the HTC slice, in
 926 the region of saturation of Network. Similarly, the bounds
 927 calculated on the blocking probability, shown in Fig. 16, are
 928 quite tight for HTC and the MTC with less reserved resources,
 929 while being looser for the other MTC slice. This behavior is
 930 due to the aggressive access to shared resources by HTC only,
 931 which makes shared resources highly utilized. In contrast,
 932 MTC slices do not overload the Network subsystem even
 933 when they have little dedicated resources. However, shared
 934 positions are used by all slices. Note in fact that when the
 935 HTC RACH throughput reaches its peak for the HTC slice
 936 (not shown in the figure, it can be however intuitively iden-
 937 tified at around 1 Gb/s from the shape of the throughput

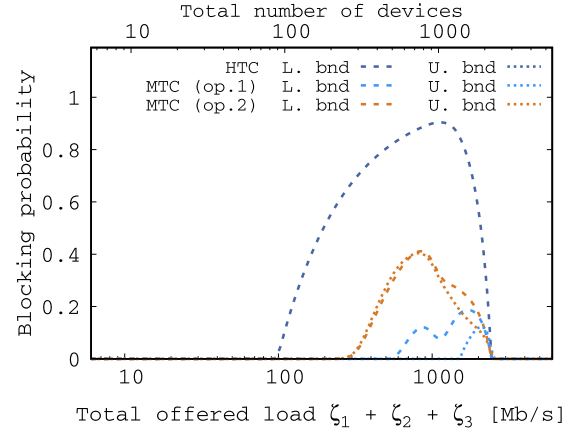


Fig. 16. Blocking probability bounds for the three-slice scenario of Table IV with 2.5, 1.5, and 1 Mb/s of capacity assigned to the three slices (i.e., 95 Mb/s of shared capacity) versus the total load in Mb/s (lower x-axis) and total number of users (upper x-axis); MTC incurs no blocking.

before and after the flat zone imposed by the Network),
 938 the pressure of HTC diminishes, and MTC slices obtain more
 939 resources. Although they do not saturate the Network sub-
 940 system throughput, still this behavior indicates that MTC devices
 941 use more shared positions, which are the ones freed by HTC
 942 after the RACH subsystem cuts off. The total population that
 943 can be served in this mixed slice case with little dedicated
 944 resources is quite low, of the order of 200 devices (about 180
 945 for HTC and 20 for MTC) before the Network throughput
 946 saturates for all slices. This means that a large pool of shared
 947

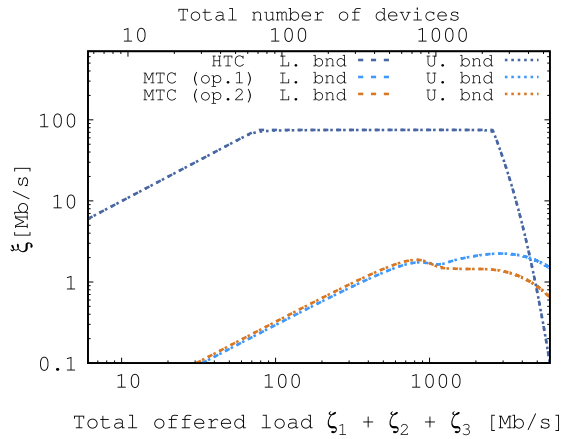


Fig. 17. Network throughput bounds for the three-slice scenario of Table IV with 25, 15, and 10 Mb/s of capacity assigned to the three slices (i.e., 50 Mb/s of shared capacity) versus the total load in Mb/s (lower x-axis) and total number of users (upper x-axis).

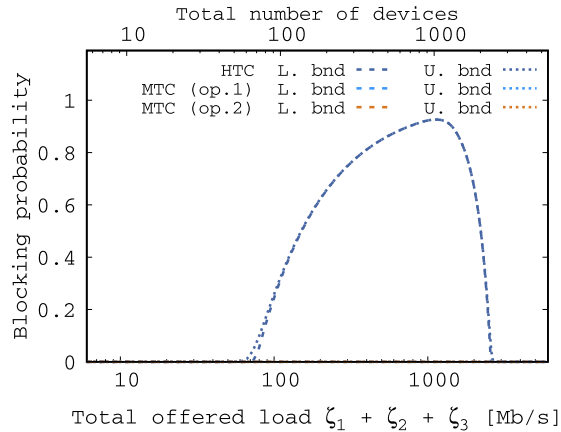


Fig. 18. Blocking probability bounds for the three-slice scenario of Table IV with 25, 15, and 10 Mb/s of capacity assigned to the three slices (i.e., 50 Mb/s of shared capacity) versus the total load in Mb/s (lower x-axis) and total number of users (upper x-axis); MTC incurs no blocking.

resources has negative impact on the size of the sustainable population.

If we now increase the amount of bandwidth allocated to slices to 25, 15 and 10 Mb/s for the HTC, MTC (operator 1) and MTC (operator 2) slices, respectively, we obtain the results shown in Figs. 17 and 18. We can see that, as expected, by reducing the amount of shared resources the bounds have become even tighter than before. In addition, we can observe an improvement in the performance of the two MTC slices, both in throughput and loss probability, which is natural, since we have increased the amount of resources for their exclusive use. In particular, no loss is now observed for MTC traffic. This comes at the cost of a slight reduction in the performance of HTC traffic, in terms of both saturation throughput and number of users. This reduction is better appreciated in Fig. 19, where we show the throughput of the HTC slice with 95, 50 and 1 Mb/s of shared capacity (while the rest of parameters remain unchanged, see Table IV). We clearly see that a reduction of the shared capacity penalizes the HTC slice, whose saturation throughput decreases from 97.5

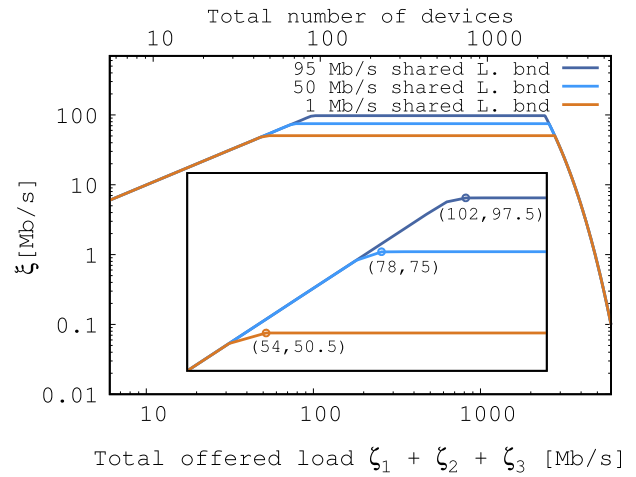


Fig. 19. Network throughput lower bounds (upper bounds are undistinguishable) for the HTC slice in the three-slice scenario of Table IV with different amounts of shared capacity (i.e., 95, 50, and 1 Mb/s) versus the total load in Mb/s (lower x-axis) and total number of users (upper x-axis). The inset shows the portion of the curves where the network throughput saturates.

to 75 and to 50.5 Mb/s. In the first case the decrease brings a benefit to the two MTC slices, at least in terms of loss probability. Instead, the second decrease does not impact the MTC slices performance. This implies that in the case of several competing slices, the choice of the amount of resources to allocate to individual slices, and consequently of the amount of resources shared among them, is a critical management issue. Too many statically allocated resources imply less flexibility, and penalize the high traffic slices. On the other hand, too many shared resources penalize the low traffic slices, because the high traffic slices tend to grab most of the shared bandwidth. A careful balance is necessary, which depends on the specific slice configuration, and requires simple but accurate models, like the one we presented in this article, to predict the performance outcome of the resource partitioning.

3) *A Six-Slice Scenario*: So far, we have observed a very good accuracy of our proposed bounds in many tested configurations. This allows us to conclude that the bounds offer a good approximation for the case of three slices. However, this does not exclude a different behavior for a higher number of slices. Thus, in order to show the ability of our approach to cope with more challenging cases, Fig. 20 shows the throughput for a case with six slices, with the configuration shown in Table V, and with the data volume of requests associated to HTC and MTC traffic specified for the previously described experiments.

This scenario could represent the case of a cell covering a city hospital and its neighborhood. Here we selected two HTC slices (e.g., one for public access and one reserved to the personnel of the hospital) plus four MTC slices (e.g., one for a factory, one for generic IoT, one for coordinating vehicles in the neighborhood, and one for hospital devices). The slices are very heterogeneous and they are assigned realistic amounts of dedicated resources, in a very heterogeneous way. Shared resources are limited in terms of capacity, while they are more generous in terms of RACH preambles and service positions in Network. Here, one HTC slice saturates

TABLE V
CONFIGURATION PARAMETERS FOR THE CASE OF SIX SLICES

Slice	Access request share	Population share	Capacity [Mb/s]	Preambles	Positions	Timeout [s]	Duty Cycle [s]
HTC (public)	0.27	0.3985	10	5	10	5	1
MTC (factory)	0.15	0.0221	20	10	20	0.1	0.1
MTC (IoT)	0.01	0.1476	0	0	0	10	10
MTC (vehicular)	0.15	0.0221	20	10	30	0.01	0.01
HTC (hospital)	0.27	0.3985	10	10	10	5	1
MTC (hospital)	0.15	0.0111	20	10	40	0.05	0.05
Shared	N/A	N/A	20	8	90	N/A	N/A

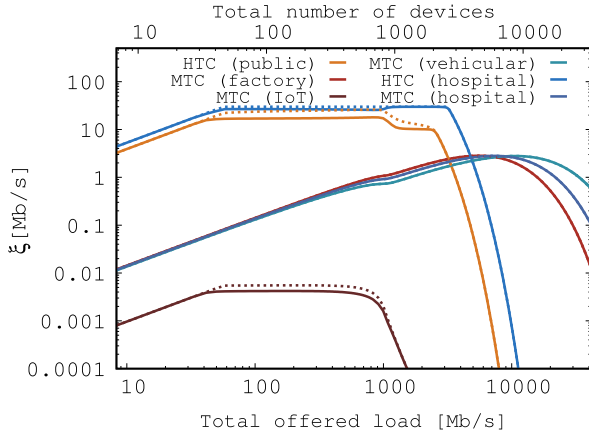


Fig. 20. Network throughput bounds for the six-slice scenario of Table V (Solid lines for lower bounds - dashed lines for upper bounds) vs. the total load (in Mb/s, lower x-axis) and total number of users (upper x-axis).

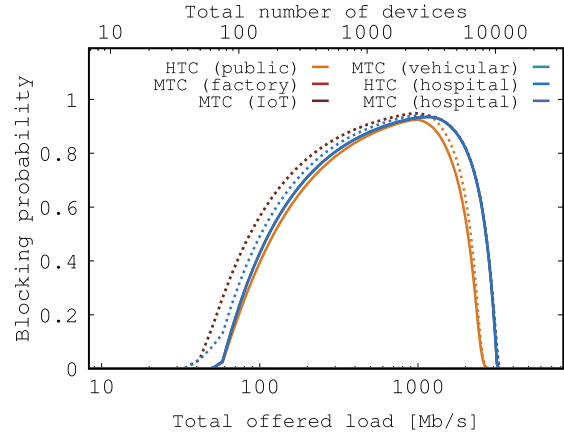


Fig. 21. Blocking probability bounds for the six-slice scenario of Table V (Solid lines for upper bounds - dashed lines for lower bounds) vs. the total load (in Mb/s, lower x-axis) and total number of users (upper x-axis).

1005 before the other because of the different number of reserved
 1006 RACH preambles, being the rest of parameters for HTC slices
 1007 the same. MTC slices are basically of two kinds. Three of
 1008 the four MTC slices are access-intensive and have large por-
 1009 tions of capacity, preambles and positions reserved for their
 1010 exclusive use. They behave similarly, although experiencing a
 1011 different cutoff, due to different numbers of dedicated posi-
 1012 tions. However, one more MTC slice has very little traffic
 1013 and no dedicated resources. For this slice, we observe satura-
 1014 tion effects at Network. Although here the MTC population
 1015 covers about 20% of the total, most of the BS resources are
 1016 dedicated, which allows to serve up to 500 devices before the
 1017 MTC slice without dedicated resources collapses.

1018 As in the cases with three slices, here we observe non-
 1019 coincident bounds only in the regions in which slices saturate
 1020 on Network. In all cases, bounds are close enough to
 1021 provide decent approximations. Similarly, Fig. 21 shows close
 1022 bounds for the blocking probability of HTC slices and for the
 1023 MTC slice with no dedicated resources (other slices incur no
 1024 blocking).

1025 Note that bounds are very important because, although on
 1026 the one hand they introduce some approximation, on the other
 1027 hand they allow the analysis to scale to otherwise unfeasible
 1028 cases. For instance, with the above discussed six-slice config-
 1029 uration, it took less than an hour to compute the 590 points
 1030 per bound per slice (about 7k points in total) that compose

Fig. 20, while it took more than a week to solve the CTMC 1031
 for the much simpler case with three slices, and it was not 1032
 conceivable to use the CTMC for the case of six slices with 1033
 a machine equipped with an Intel Xeon CPU E5-1620 v3 @ 1034
 3.50GHz. 1035

Besides good accuracy of the bounds, the figures for sce- 1036
 narios with multiple slices show that the interaction between 1037
 slices becomes evident only when one or more slices, but not 1038
 all of them, experience a bottleneck in Network and the 1039
 RACH drops requests aggressively. This has negative impact on 1040
 the affected slices but also frees resources for other slices (see, 1041
 e.g., the changes of slope for the throughput with more than 1042
 2 Gb/s in Fig. 15 and for more than 1 Gb/s in Fig. 20). Most 1043
 importantly, some slices can incur RACH or Network bottle- 1044
 necks well before other slices, so that it is not easy to correctly 1045
 dimension the resources to be allocated to each slice. However, 1046
 our model, with the bounding technique we propose, offers a 1047
 powerful tool to identify and test suitable configurations with 1048
 limited computational complexity. 1049

V. RELATED WORK 1050

An overview of network slicing concepts, architectures and 1051
 algorithms was recently provided by two special issues of 1052
 the IEEE Communications Magazine [12], [13]. Moreover, 1053
 network slicing in the RAN and heterogeneous traffic types 1054
 are being investigated under different perspectives, although 1055

not yet from the point of view of their compound requirements and interactions.

Resource allocation is one of the key challenges to tackle, and, accordingly, a number of proposals are sprouting these days. For instance, the authors of [14] propose an orchestration system that leverages deep learning techniques, so as to follow traffic fluctuations and allocate resources to slices accordingly. 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. More practical studies like [16] show via simulation that earliest deadline first (EDF) scheduling represents a practical and effective solution for performance isolation with dynamic resource allocation in RAN slicing scenarios. The optimal allocation of resources to slices is addressed in [17], where a distributed algorithm is 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. Furthermore, the authors of [18] show that physical transmission resources can be sliced using millimeter wave techniques, while the authors of [19] demonstrate that physical resources assigned to slices need to be coordinated across multiple cells, otherwise slices cannot fully exploit the properties of physical level protocols. Other studies show the importance of per-slice resource allocation to satisfy non-trivial performance indicators [20]. There are also active initiatives devoted to develop concepts and implementations of network slicing, e.g., the European Commission-founded projects 5G-CROSSHAUL [21], 5G-TRANSFORMER [22], and 5G-NORMA [23]. However, the existing approaches somehow neglect the role of network access procedures, which, as we have shown in this article, can introduce unexpected behavioral trends in the access network.

Besides resource allocation, there are several works pointing at performance issues of network slicing, and proposing optimization schemes, e.g., in [24], a dynamic RAN cell slicing controller was proposed and evaluated by simulation in a urban setting comprising 19 microcells, showing that the proposed controller performs better than a distributed static slicing solution and a centralized load balancing solution. The authors of [25] further present radio slicing implementation with 5G NR and discuss potential slice configurations, while the use of machine learning to manage the resources of 5G radio slices is discussed in [26]. An optimization problem for radio resource sharing among slices in a cell is studied in [27], 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 [28]. Simulation results show that the proposed scheme achieves better performance than baseline schemes. The authors of [29] propose to use a Cross Layer Controller to orchestrate SDN and SDR technologies, so to unify the control of radio and

transport protocols; they use simulations to show that significant gain stems from the coordination of slicing in different network segments. The analysis of the market composed by one infrastructure provider and several tenants that rent a network slice to provide service to their customers is tackled in [30]. 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 proposed algorithm is evaluated by simulation. The sharing of resources among slices is investigated in [31]. Each slice is assigned a fixed portion of 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 admits a Nash equilibrium. The effectiveness of the proposed solution is studied 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 [32]. 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. This body of work is important, although we claim that the techniques proposed in there should be revisited to take in consideration the presence of potential RACH bottlenecks. Our model could be used to enable such study.

More specifically, 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, which allow for the derivation of exact through computationally complex expressions for the key performance indicators, given a slice-set configuration. They also allow to derive non-trivial tight bounds for blocking probability and throughput, which scale efficiently in the analysis of several slices. Note that, in our case, simulation just serves the purpose of validating the accuracy of the analytical model. Note also that our results are not meant to contrast the findings of other works, neither they are proposing novel resource management methods. Instead, our model and bounds shed light on the intricacies of RAN sharing mechanisms in an interpretable manner, and are instrumental in enhancing and speeding up resource management optimization tools like the ones mentioned in this section.

VI. CONCLUSION

In this article we described a detailed stochastic model of the behavior of radio access in a sliced RAN 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 typical cell comprising one HTC and one MTC slice, we observed the mutual effects of slice traffic increases on performance, exposing unexpected

behaviors for the traffic values at which the RACH is close to saturation. With more slices, the interaction becomes more cumbersome to predict and more complex to evaluate in detail. However, efficient bounds allow to scale the analysis up to several heterogeneous slices with limited computational power.

We have studied a cell with standard 5G configurations for what concerns the RACH and the number of service positions at the base station. With these parameters, we have shown that current technologies allow to sustain a few hundreds of HTC devices and a several thousands of MTC devices, which calls for protocol enhancements in order to scale 5G networks to more massive use cases.

REFERENCES

- [1] V. Mancuso, P. Castagno, M. Sereno, and M. A. Marsan, "Slicing cell resources: The case of HTC and MTC coexistence," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2019, pp. 667–675.
- [2] (2011). *The NGMN Alliance—At a Glance*. [Online]. Available: <http://www.ngmn.org>
- [3] The NGMN Alliance. (Jan. 2016) *Description of Network Slicing Concept*. [Online]. Available: https://www.ngmn.org/uploads/media/160113_Network_Slicing_v1_0.pdf
- [4] *System Architecture for the 5G System, Version 15.2.0 - Release 15*, 3GPP Standard TS 23.501, 2018. [Online]. Available: https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/15.02.00_60/ts_123501v150200p.pdf
- [5] P. Castagno, V. Mancuso, M. Sereno, and M. A. Marsan, "A simple model of MTC in smart factories," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, 2018, pp. 2591–2599.
- [6] M. B. Shahab, R. Abbas, M. Shirvanimoghaddam, and S. J. Johnson, "Grant-free non-orthogonal multiple access for IoT: A survey," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 3, pp. 1805–1838, 3rd Quart., 2020.
- [7] S. Ali, N. Rajatheva, and W. Saad, "Fast uplink grant for machine type communications: Challenges and opportunities," *IEEE Commun. Mag.*, vol. 57, no. 3, pp. 97–103, May 2019.
- [8] "Technical specification group radio access network; study on RAN improvements for machine-type communications, release 11," 3GPP, Sophia Antipolis, France, Rep. TR 37.868, 2011.
- [9] F. Baskett, K. M. Chandy, R. R. Muntz, and F. G. Palacios, "Open, closed, and mixed networks of queues with different classes of customers," *J. ACM*, vol. 22, no. 2, pp. 248–260, Apr. 1975.
- [10] G. Ciardo, R. L. Jones, III, A. S. Miner, and R. Siminiceanu, "Logical and stochastic modeling with smart," in *Proc. Comput. Perform. Eval. Model. Techn. Tools*, 2003, pp. 78–97.
- [11] M. Sereno, "Computational algorithms for product-form of competing Markov chains," in *Proc. 10th Int. Workshop Petri Nets Perform. Models (PNPM)*, 2003, pp. 93–102.
- [12] K. Samdanis *et al.*, "5G network slicing—Part 1: Concepts, principles, and architectures," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 70–71, May 2017.
- [13] K. Samdanis *et al.*, "5G network slicing—Part 2: Algorithms and practice," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 110–111, Aug. 2017.
- [14] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "AZTEC: Anticipatory capacity allocation for zero-touch network slicing," in *Proc. IEEE INFOCOM*, 2020, pp. 794–803.
- [15] P. Caballero *et al.*, "Network slicing games: Enabling customization in multi-tenant networks," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
- [16] T. Guo and A. Suarez, "Enabling 5G RAN slicing with EDF slice scheduling," *IEEE Trans. Veh. Technol.*, vol. 68, no. 3, pp. 2865–2877, Mar. 2019.
- [17] P. Caballero *et al.*, "Multi-tenant radio access network slicing: Statistical multiplexing of spatial loads," *IEEE/ACM Trans. Netw.*, vol. 25, no. 5, pp. 3044–3058, Sep. 2017.
- [18] M. Pagin, F. Agostini, T. Zugno, M. Polese, and M. Zorzi, "Enabling RAN slicing through carrier aggregation in mmWave cellular networks," in *Proc. MedComNet*, 2020, Art. no. 10142.
- [19] S. D'Oro *et al.*, "The slice is served: Enforcing radio access network slicing in virtualized 5G systems," in *Proc. IEEE INFOCOM*, 2019, pp. 442–450.
- [20] J. Martin-Perez, F. Malandrino, C.-F. Chiasserini, and C. J. Bernardos, "OKpi: All-KPI network slicing through efficient resource allocation," in *Proc. IEEE INFOCOM*, 2020, pp. 804–813.

- [21] X. Li *et al.*, "5G-crosshaul network slicing: Enabling multi-tenancy in mobile transport networks," *IEEE Commun. Mag.*, vol. 55, no. 8, pp. 128–137, May 2017.
- [22] C. Casetti *et al.*, "Network slices for vertical industries," in *Proc. IEEE Wireless Commun. Netw. Conf. Workshops (WCNCW)*, 2018, pp. 254–259.
- [23] F. Z. Yousaf *et al.*, "Network slicing with flexible mobility and QoS/QoE support for 5G networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC Workshops)*, 2017, pp. 1195–1201.
- [24] P. C. Garces, X. Costa-Perez, K. Samdanis, and A. Banchs, "RMSC: A cell slicing controller for virtualized multi-tenant mobile networks," in *Proc. IEEE 81st VTC Spring*, 2015, pp. 1–6.
- [25] S. E. Elayoubi, S. B. Jemaa, Z. Altman, and A. Galindo-Serrano, "5G RAN slicing for verticals: Enablers and challenges," *IEEE Commun. Mag.*, vol. 57, no. 1, pp. 28–34, Jan. 2019.
- [26] Y. Hua, R. Li, Z. Zhao, X. Chen, and H. Zhang, "GAN-powered deep distributional reinforcement learning for resource management in network slicing," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 334–349, Oct. 2020.
- [27] C.-Y. Chang, N. Nikaein, and T. Spyropoulos, "Radio access network resource slicing for flexible service execution," in *Proc. IEEE INFOCOM WKSHPs*, 2018, pp. 668–673.
- [28] Y. L. Lee *et al.*, "Dynamic network slicing for multitenant heterogeneous cloud radio access networks," *IEEE Trans. Wireless Commun.*, vol. 17, no. 4, pp. 2146–2161, Jan. 2018.
- [29] S. Barmounakis, N. Maroulis, M. Papadakis, G. Tsiatsios, D. Soukaras, and N. Alonistioti, "Network slicing—Enabled RAN management for 5G: Cross layer control based on SDN and SDR," *Comput. Netw.*, vol. 166, Jan. 2020, Art. no. 106987. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1389128619309776>
- [30] D. Bega *et al.*, "Optimising 5G infrastructure markets: The business of network slicing," in *Proc. IEEE INFOCOM*, 2017, pp. 1–9.
- [31] J. Zheng *et al.*, "Statistical multiplexing and traffic shaping games for network slicing," in *Proc. 15th WiOpt*, 2017, pp. 1–8.
- [32] D. Nojima *et al.*, "Resource isolation in RAN part while utilizing ordinary scheduling algorithm for network slicing," in *Proc. IEEE 87th Veh. Technol. Conf. (VTC Spring)*, 2018, pp. 1–6.



Vincenzo Mancuso received the Ph.D. degree from the University of Palermo, Italy, in 2005. He is currently a Research Associate Professor with IMDEA Networks, Madrid, Spain, and a recipient with the Ramon y Cajal Research Grant, Spanish Ministry of Science and Innovation. He was with INRIA, France; Rice University, USA; and the University of Palermo, Italy. His research focus is on analysis, design, and experimental evaluation of opportunistic wireless architectures, and mobile broadband services.



Paolo Castagno received the master's and Ph.D. degrees from the University of Torino, in 2014 and 2018, respectively. He is currently a Postdoctoral Researcher with the Department of Computer Science, University of Torino, Italy. His research focus is on performance, evaluation of computer systems, and communication networks, with a specific interest on wireless networks.

1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308



Matteo Sereno was born in Nocera Inferiore, Italy. He received the Laurea degree in computer science from the University of Salerno, in 1987, and the Ph.D. degree in computer science from the University of Torino, in 1992. He is currently a Full Professor with the Computer Science Department, University of Torino. His current research interests are in the area of performance, evaluation of computer systems, communication networks, peer-to-peer systems, compressive sensing and coding techniques in distributed applications, game theory, queueing networks, and stochastic Petri net models.



Marco Ajmone Marsan (Life Fellow, IEEE) received the first graduation degree in EE from the Politecnico di Torino, in 1974, the second graduation degree in EE from the University of California, Los Angeles, in 1978, and the Honorary Doctoral degree from the Budapest University of Technology and Economics, in 2002. He is currently a Full Professor with the Department of Electronics and Telecommunications, Politecnico di Torino, Italy, and a part-time Research Professor with IMDEA Networks Institute, Leganes, Spain. Since 1974, he has been with the Politecnico di Torino, in different roles of an academic career, with an interruption from 1987 to 1990, when he was a Full Professor with the Department of Computer Science, University of Milan. He has been doing research in the fields of digital transmission, distributed systems, and networking. He has been a Member of the Editorial Board and the Steering Committee of the ACM/IEEE TRANSACTIONS ON NETWORKING. He is a Member of the Editorial Boards of the journals *Computer Networks* and *Performance Evaluation* (Elsevier), and the *ACM Transactions on Modeling and Performance Evaluation of Computer Systems*. He served in the organizing committee of several leading networking conferences, and he was the General Chair of INFOCOM 2013. He is a Member of the Academy of Sciences, Torino, and a Member of Academia Europaea.

1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331