ACCEPTED FROM OPEN CALL

Design and Analysis of 5G Scenarios with `simmer`: An R Package for Fast DES Prototyping

Iñaki Ucar, José Alberto Hernández, Pablo Serrano, and Arturo Azcorra

The authors present an analysis of three 5G scenarios using simmer, a recent R package for discrete-event simulation that sits between the above two paradigms. As their results show, it provides a simple yet very powerful syntax, supporting the efficient simulation of relatively complex scenarios at a low implementation cost.

Abstract

Simulation frameworks are important tools for the analysis and design of communication networks and protocols, but they can be extremely costly and/or complex (for the case of very specialized tools), or too naive and lacking proper features and support (for the case of ad-hoc tools). In this article, we present an analysis of three 5G scenarios using `simmer`, a recent R package for discrete-event simulation that sits between the above two paradigms. As our results show, it provides a simple yet very powerful syntax, supporting the efficient simulation of relatively complex scenarios at a low implementation cost.

Introduction

Simulation frameworks are undoubtedly one of the most important tools for the analysis and design of communication networks and protocols. Their applications are numerous, including the performance evaluation of existing or novel proposals, dimensioning of resources and capacity planning, or the validation of theoretical analyses, which are based on simplifying assumptions whose impact is to be assessed.

In fact, simulation frameworks also make a number of simplifying assumptions, typically about components of the considered system that are not directly related to the performance variable of interest, to reduce complexity so the development of the scenario is easier and numerical figures are obtained faster. This "complexity" axis goes from very specialized, large simulation tools such as NS-3, OMNeT++, OPNET to ad-hoc simulation tools, consisting of hundreds of lines of code, typically used to validate a very specific part of the network or a given mathematical analysis. (The list of network simulation tools is vast; see e.g., http://people.idsia.ch/~andrea/sim/simnet.html, accessed on Feb. 2018.) The latter are often developed over general-purpose languages such as C/C++ or Python, over numerical frameworks such as Matlab, or over some framework for discrete-event simulation. See e.g., https://en.wikipedia.org/wiki/Listofdiscreteeventsimulationsoftware (accessed on Feb. 2018).

On the one hand, the complexity of specialized tools (such as their cost, if applicable) preclude their use for short-to-medium research projects, as the learning curve is typically steep, plus they are difficult to extend, which is mandatory to test a novel functionality. On the other hand, the development of ad-hoc tools also requires some investment of time and resources, lack a proper validation of their functionality, and furthermore, there is no code maintenance once the project is finished, for the few cases in which the code is made publicly available.

In this article, we introduce the use of a recent event-driven simulation package, `simmer`, and show its applicability in fast prototyping three 5G-related scenarios. `simmer` sits between the above two complexity extremes, and combines a number of features that support, among others, versatility and repeatability. More specifically, some of the key advantages of `simmer` are as follows:

- It is based on the very popular R programming language, which benefits from a large community of users and contributors, but also natively supports the analysis of results via the many R statistical and visualization packages.
- The code has been peer-reviewed [1], and it is an official package, with numerous examples readily available, potentially supported by a notable user population.
- In addition to its ease of use and versatility, its code is partially optimized for speed, and therefore it can simulate relatively complex scenarios under reasonable times.

We illustrate the use of `simmer` by simulating three different networking scenarios, which are inspired by current research trends regarding the design of fifth-generation (5G) mobile networks [2]. These diverse scenarios confirm the validity of `simmer` as a useful simulation tool that can support (at least as a first step) the dimensioning of communication systems, or can serve to quantify the trade-offs imposed by a given technology decision. More specifically, we consider the analysis of the following scenarios:

- Different design options for a *crosshaul* scenario, where packetized in-phase and quadrature samples from fronthaul traffic are transmitted along backhaul data frames over the same links. Thanks to `simmer` and its support for statistical analysis, we can easily quantify delays under several queueing disci-

Digital Object Identifier:
10.1109/MCOM.2018.1700960

The authors are with Universidad Carlos III de Madrid; Arturo Azcorra is also with IMDEA Networks Institute.

0163-6804/18/$25.00 © 2018 IEEE

plines for different fronthaul/backhaul ratios, these being the metrics of interest for these scenarios.

- The impact of installing small cells in a fiber-to-the-premises scenario. Here, we analyze two different approaches to support the highly-demanding cellular traffic along with the existing residential traffic, namely, the deployment of a remote radio head vs. the deployment of a small cell.
- Massive Internet-of-Things scenarios, where thousands of metering devices share the same channel to upload their readings. Here, we analyze the impact of access parameters on performance, with a particular interest in the energy required to deliver the information, which will ultimately impact the lifetime of devices running on batteries.

This article provides a quick overview of `simmer` and its key features. The analyses of the three considered 5G-related scenarios showcase the multiple benefits of different approaches, and the versatility of `simmer` to easily implement their key features.

An Introduction to `simmer`

`simmer` (available for download on the Comprehensive R Archive Network (CRAN), https://CRAN.R-project.org/package=simmer, accessed on Feb. 2018) [1] is a discrete event simulation (DES) library for the R Project (https://www.R-project.org/, accessed on Feb. 2018), the open source programming language for statistical computing that has been receiving increased attention, primarily due to its widespread adoption for data science, analytics, and statistical research.

By developing `simmer` for R, it can benefit from this growing ecosystem. Note that `simmer` is not intended to be a substitute for NS-3 or OMNeT++, which are the de facto standards for open-source network simulations. Instead, `simmer` is designed as a general-purpose DES framework with a human-friendly syntax, and a very gentle learning curve. It can be used to complement other field-specific simulators as a rapid prototyping tool that enables insightful analysis of different designs. As we will illustrate in the next section, with `simmer` it is simple to simulate relatively complex scenarios, with the added benefit of the availability of many convenient data analysis and representation libraries, thanks to the use of R.

The R application programming interface (API) exposed by `simmer` revolves around the concept of *trajectory*, which defines the "path" in the simulation for entities of the same type. A trajectory is a recipe for the arrivals attached to it, an ordered set of actions (or *verbs*) chained together with the pipe operator (%>%, whose behavior is similar to the command-line pipe). The following example illustrates a basic `simmer` workflow, modeling the classic case of customers being attended by a single clerk with infinite waiting space in a few lines of code:

```
1 library(simmer)

3 cust <- trajectory("customer") %>%
    seize("clerk", amount=1) %>%
5   timeout(function() rexp(1, 2)) %>%
    release("clerk", amount=1)
7
  env <- simmer("bank") %>%
9   add_resource("clerk", capacity=1,
      queue_size=Inf) %>%
    add_generator("cust", cust,
      function() rexp(1, 1)) %>%
11  run(until=1000)

13  arrivals <- get_mon_arrivals(env)
    resources <- get_mon_resources(env)
```

Given that both the time at the clerk and between customers are exponential random variables, and the infinite queue length, this example corresponds, in Kendall's notation, to an M/M/1 queue. It serves to illustrate the two main elements of `simmer`: the `trajectory` object and the `simmer` environment (or *simulation environment*).

The `customer` trajectory (line 3) defines the behavior of a generic customer: seize a clerk, spend some time, and release it. The `env` simulation environment (line 8) is then defined as one clerk with infinite queue size and a generator of customers, each one following the trajectory defined above. Based on this syntax, the flexibility is provided through a rich set of activities (more than 30) that can be appended to trajectories, which support changing arrivals' properties (attributes, priority, batches); different interactions with the resources (select, seize, release, change their properties); and the generators (activate, deactivate, change their properties), and even the definition of branches (simple, depending on a condition, or parallel) and loops. Finally, some support for asynchronous programming is also provided (subscription to signals and registration of handlers).

In addition to providing a powerful yet simple syntax, `simmer` is also *fast*, for example, faster than equivalent frameworks such as SimPy and SimJulia for the Python and Julia languages, respectively [1]. The key for this speed is its underlying simulation core, which is written in C++. Furthermore, and perhaps more importantly, `simmer` implements automatic monitoring capabilities: every event is accounted for by default, both for arrivals (starting and ending times, activity time, ending condition, resources traversed) and resources (server and queue status), and all this information can be easily retrieved in standard R data frames for further processing of results (lines 13-14 of the *clerk* example).

`simmer` implements automatic monitoring capabilities: every event is accounted for by default, both for arrivals (starting and ending times, activity time, ending condition, resources traversed) and resources (server and queue status), and all this information can be easily retrieved in standard R data frames for further processing of results (lines 13-14 of the clerk example).

Modeling 5G Scenarios

In what follows, we will model and analyze three representative 5G scenarios using `simmer`. The source code for the use cases presented here, including configuration (definition of constants and parameters), simulation and analysis of results, is available online (see the "Articles" section at http://r-simmer.org, and the GitHub repository at https://github.com/r-simmer/simmer, accessed on Feb. 2018), while summary statistics of the simulations performed and their complexity are provided below.

Crosshauling of FH and BH Traffic

This scenario is motivated by the Cloud Radio Access Network (C-RAN) paradigm [3], where the mobile base station functionality is split into

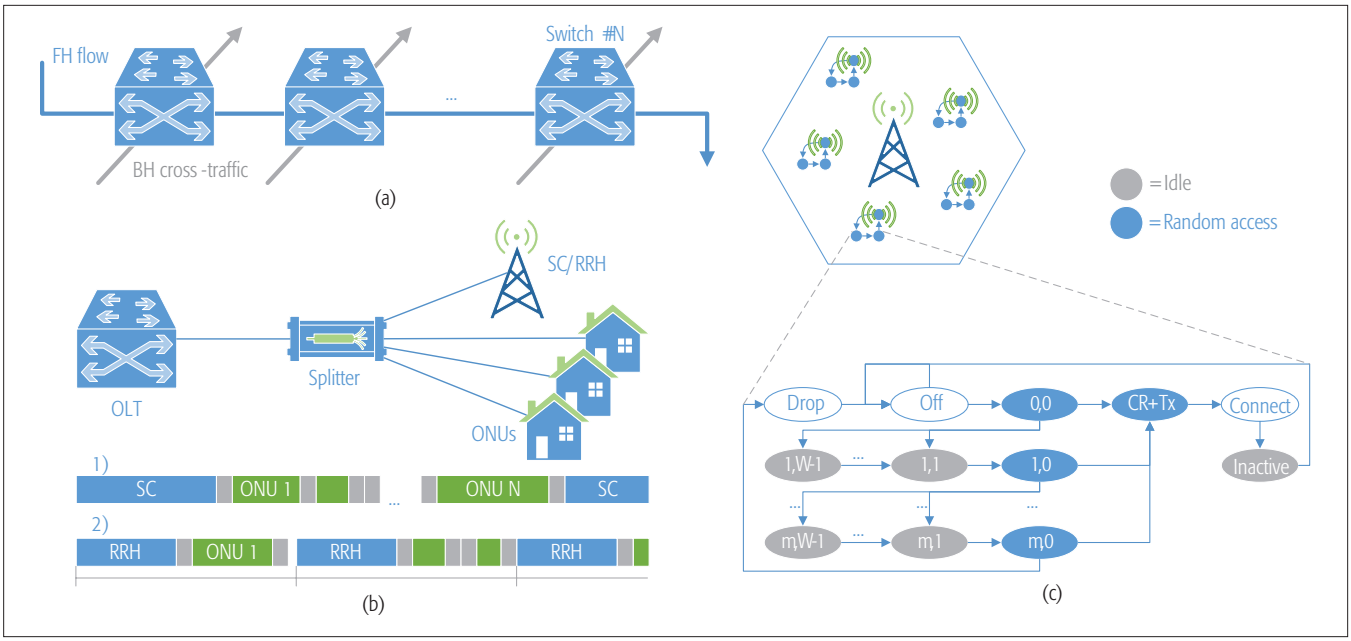

Figure 1. Use cases: description of: a) scenario no. 1; b) scenario no. 2; c) scenario no. 3.

simple remote radio heads (RRH), spread across the deployment, and connected by fiber to centralized (and possibly virtualized) base-band units (BBU), at the operators' premises. C-RAN is an architectural shift, aiming at providing capital and operational expenditure savings while supporting better interference reduction, and improved performance via Coordinated Multi-Point (CoMP).

In this C-RAN paradigm, fronthaul (FH) traffic from the RRH has stringent delay requirements, while backhaul (BH) traffic from the BBU has mild delay requirements. In a general topology, such as the one illustrated in Fig. 1a ("scenario no. 1") [4], packet switches will forward both types of traffic, and therefore introducing service differentiation might improve the ability to fulfil the delivery guarantees of FH traffic, which is in-line with the IEEE 802.1CM Time-Sensitive Networking for Fronthaul standard under development (http://www.ieee802.org/1/pages/802.1cm.html, Draft 0.6, accessed on Feb. 2018). To quantify the benefits of this differentiation, we use `simmer` to simulate the scenario, thus supporting, for example, a design decision about the best scheme to deploy.

We assume that packet switches are at 75 percent load, and operate at 40 Gb/s line rate. We assume 50 percent of this load corresponds to FH traffic, and the other 50 percent corresponds to BH traffic. Packet arrivals follow a Poisson process (although any other arrival process may be used in the simulation, for instance, bursty, self-similar). Regarding packet sizes, FH packets are assumed to be CPRI (Common Public Radio Interface) basic frames of 80 byte length (i.e., CPRI option 4, see [5]), while BH packets follow the classic AMS-IX (Amsterdam Internet Exchange) trimodal function, namely seven out of 12 packets are short (40 bytes), four out of 12 are medium size (576 bytes), and one out of 12 packets is long (1500 bytes) (Amsterdam Internet Exchange Ethernet Frame Size Distribution, statistics available online at https://ams-ix.net/technical/statistics/sflow-stats/frame-size-distribution, accessed on Feb. 2018).

We consider three different policies for service differentiation at the N switches:

- No service differentiation.
- Strict priority (SP) is given to FH over BH, but without preemption.
- SP is given to FH over BH with preemption.

For each of these policies, we first simulate a one-switch scenario with the traffic characteristics described above, and compute the queueing delay for FH and BH traffic. The results are depicted in Fig. 2a, where we use box-plots to illustrate the {5, 25, 50, 75, 95}-th percentiles.

As the figure shows, the first strategy ("without SP") results in both types of traffic experiencing the same queueing delay. The second strategy ("with SP"), that is, service differentiation without preemption, results in a much improved service for FH traffic, as FH packets only have to wait for other FH packets ahead, and sometimes one BH packet being served (i.e., the residual service time). The third strategy ('with SP & preemption') implements a preemption strategy, discarding BH traffic from the transmission line if a FH packet arrives which, as the figure shows, does significantly improve delay performance (the 95th percentile drastically decreases), which was caused by the long transmission times of long BH frames (1500 bytes). For this scenario, we conclude that a preemption strategy reduces queuing delay to the minimum, with very high delivery guarantees.

We next analyze scenarios where the FH has to transverse N packet switches in tandem, each one also serving BH traffic, as illustrated in Fig. 1a. We depict in Fig. 2b the queueing delays of FH traffic for N = {1, 2, 5}, under the three considered policies (the N = 1 case corresponds to the same results as in the previous experiments). As shown, FH queueing delay accumulates after traversing multiple switches for the first and second policies, both in terms of median and percentiles, hence jitter too. Only the SP strategy with preemption keeps both delay and jitter values extremely low, since FH packets need to wait only if the server is

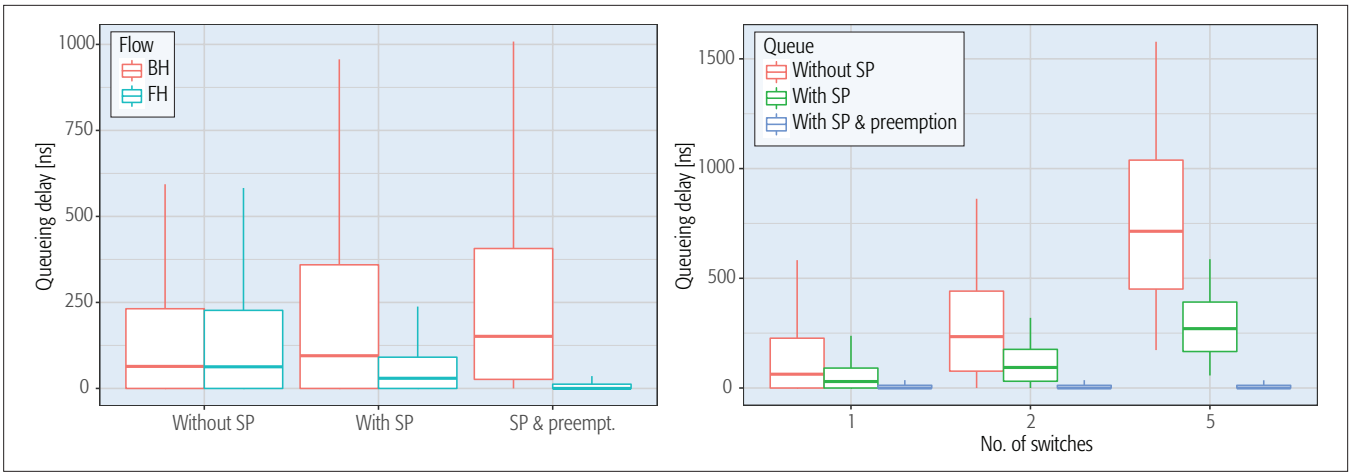

Figure 2. Queueing delay experienced by FH and BH traffic under different QoS policies: a) FH/BH queueing delay comparison for a single switch; b) accumulated FH delay for several packet switches. The whiskers represent the 5th and 95th percentiles.

occupied by other FH packets. In contrast, the SP strategy without preemption shows accumulated packet delay and packet delay variability after traversing multiple switches.

Implementation Details: The simulation implements a single trajectory for the FH traffic, `fh_traffic`, which seizes the *N* packet switches sequentially. Additionally, a list of *N* trajectories called `bh_traffic`, one for each switch, models the interfering BH traffic. The switches are defined as resources with `capacity`=1, and infinite queue length. A generator of FH traffic plus *N* generators of BH traffic are attached to their respective trajectories. All the elements are encapsulated into a function, and a number of constants are parameterized, namely, the number of switches, FH traffic's priority, and whether the switches should be preemptive or not. As can be seen, all the `cases`, defined as a data frame (one case per row), are easily parallelized using standard R tools (i.e., the R-core `parallel` package).

All the monitoring information automatically collected by `simmer` for the arrivals (BH and FH packets) can be retrieved as a data frame using the `get_mon_arrivals`() method. This enables further analysis and visualization in only a few lines of code using the `dplyr` (https://CRAN.R-project.org/package=dplyr, accessed on Feb. 2018) and `ggplot` (https://CRAN.R-project.org/package=ggplot2, accessed on Feb. 2018).} packages.

Mobile Traffic Backhauling with FTTx

We next consider the case of a residencial area with a Fiber-To-The-Premises (FTTx) infrastructure, that is, an optical distribution network (ODN), composed of the optical line terminal (OLT), splitters, and the optical network unit (ONU) at the users' premises. As Fig. 1b illustrates ("scenario no. 2"), we assume that an operator is planning to deploy an antenna, carrying the mobile traffic over the ODN, and is considering two implementation options:

- Deployment of a small cell, reducing the amount and requirements of the generated traffic.
- Deployment of an RRH, following the C-RAN paradigm discussed above, which would therefore generate time-sensitive FH traffic.

In both cases, we analyze the upstream channel of a time-division multiplexed passive optical network (TDM-PON) providing broadband access to the residential and mobile users. We assume for simplicity the case of ITU-T G.984 Gigabit PON (it would be straightforward to extend the results to other TDM-based PONs, such as XG-PON, XGS-PON, EPON, 10G-EPON), with a total capacity of 1.25 Gb/s in the upstream, and where each ONU generates 20 Mb/s of upstream traffic.

Small Cell: Here we assume that the small cell generates 150 Mb/s peak traffic, and shares the ODN with 31 residential users, which corresponds to an average total load of 61.6 percent. We assume bursty arrivals following a compound Poisson model, where both the bursts and the burst length are Poisson-distributed, with a mean of 20 packets. As in the previous use case, packet sizes are randomly chosen following the AMS-IX trimodal distribution, yielding an average burst size of 6407 Bytes. In this case, we assume a dynamic bandwidth allocation (DBA) algorithm similar to the IPACT protocol[6], where each ONU requests at the end of its transmission window resources for the next cycle, while the OLT receives such requests, and decides when and for how long each ONU may transmit its data in the next cycle granting transmission windows in a round-robin fashion. We assume a 1 ms guard-time between transmission windows of consecutive ONUs.

We consider four different quality-of-service (QoS) policies for the DBA, these being defined by the maximum supported request per cycle per ONU (the cellular traffic is always granted their requests in each case). If an ONU requests more than this maximum transmission window (namely, 1500 B, 3000 B, 6000 B or infinite, which means no limit), the OLT will only grant this maximum, and therefore the rest of the traffic will queue at the ONU.

RRH Backhauling: Next, we assume an RRH generating FH traffic, following the MAC-PHY (Media Access Control, physical layer) functional split, that is, one OFDM (Orthogonal Frequency Division Multiplexing) symbol is sampled, quantized (approx. 6000 B), and transmitted to the BBU every 66.67 ms, this resulting in an approximate rate of 720 Mb/s. This FH traffic now shares

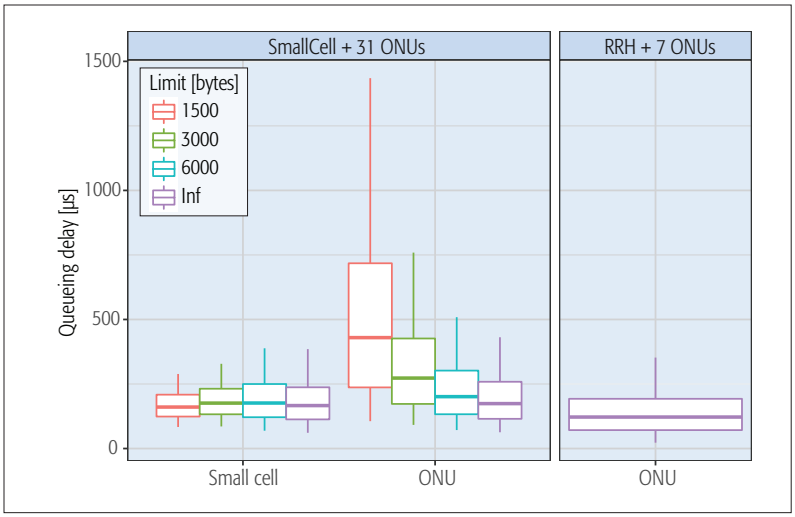

Figure 3. Upstream delay for small cell and residential users sharing the upstream channel in a TDM-PON.

the ODN with seven residential ONUs, resulting in an approximate total load of 68.8 percent. To fulfil the tight delivery requirements of FH traffic, we assume that the DBA algorithm guarantees periodic TDM reservations for the FH traffic, and then the ONUs use a similar algorithm as before to share the rest of the available bandwidth.

We depict in Fig. 3 (left) the results corresponding to the uplink delay in Case 1, that is, one small cell and 31 ONUs, for both types of traffic and the four QoS policies. As in the previous case, we focus on the queueing delay, and represent the different percentiles with box-and-whisker plots. As the figure shows, when requests are not limited ("Inf"), the traffic from the ONUs and the small cell experience the same delay, as expected. However, the moment the DBA algorithm enforces a limit to ONU requests, differences appear: the small cell delays are smaller and less disperse, while the results for the ONU are longer and show more variability. It can be seen that with a 1500 B limit, the delivery guarantees for the small cell traffic are very tight (e.g., 95th percentile around 250 µs), which is achieved at the expense of significant delays for the ONUs.

Next we depict in Fig. 3 (right) the results for the uplink delay in Case 2, that is, a RRH and seven ONUs. We do not depict here the delay results corresponding to the FH traffic, as in this case the reservation mechanism guarantees its delivery with zero queueing delay. As the figure shows, the delay performance obtained by users is slightly better than in the previous case, although fewer users can be accommodated due to traffic demands.

Implementation Details: This scenario serves to illustrate a different strategy to code a simulation with `simmer`. Instead of a trajectory attached to an unlimited generator of arrivals, the OLT is defined here as a trajectory with a single worker in an infinite loop by using the `rollback()` activity. This OLT executes the DBA logic encapsulated into the `set_next_window()` function, and timely activates the ONUs, defined as resources, in a round-robin fashion. This is achieved by modifying the capacity of a given ONU resource to meet the number of packets allocated for the next transmission window.

N generators feed traffic into *N* trajectories defined as a list of `ONUs`. Packets arriving there, similar to the previous use case, first seize the corresponding ONU resource (which acts as a *token bucket*), and as soon as the `OLT` increments the capacity, they seize the link to be transmitted. An additional ONU is defined with a different traffic rate for the small cell case. As for the RRH case, the RRH trajectory is added, which holds a single worker in a loop seizing and releasing the link periodically.

As in the previous use case, a number of constants are parameterized (scenario, number of ONUs, and TDM reservation limit), and all the combinations are parallelized with the R-core `parallel` package. Finally, the monitoring statistics for the arrivals are retrieved, and a similar analysis is made using `dplyr` and `ggplot` in very few lines of code.

Energy Efficiency for Massive IoT

Finally, we consider the case of a massive Internet-of-Things (mIoT) scenario, a use case for Long Term Evolution (LTE) and next-generation 5G networks, as defined by the Third Generation Partnership Project (3GPP) in [7]. As Fig. 1c (top) illustrates, we consider a single LTE macrocell in a dense urban area. The buildings in the cell area are populated with *N* smart meters (for electricity, gas, and water), and each meter operates independently as a Narrowband IoT (NB-IoT) device.

The devices' behavior is modeled following the diagram depicted in Fig. 1c (bottom), which is a simplified version of the Markov chain model developed in [8] (Fig. 5). A device may stay in `RRC Idle` ("Off"), and awakes with some periodicity to upload its reading. This communication phase encompasses a contention-based random access (RA) procedure, with a backoff time randomly chosen between (0, *W*) time slots, and up to *m* retransmissions. If the connection request fails, the reading is dropped, and the device returns to the "Off" state. If the connection is successful, we assume that the device implements the Control Plane Cellular IoT (CP) optimization [8], so that the data is transmitted over the RRC connection request phase using the non access stratum (NAS) level. Then, the device has to wait ("Inactive") until the connection is released, and eventually returns to the "Off" state.

The goal of this use case is to study the effect of synchronization across IoT devices (for instance, due to a power outage) in the energy consumption. As in [9], we assume that a device provides its readings as often as every hour, and the cases of $N = \{5, 10, 30\} \cdot 10^3$ devices in one cell are considered. In order to study different levels of synchronization, each node implements an additional backoff window prior to the RA procedure. Furthermore, we selected $m = 9$ and $W = 20$; the rest of the parameters (power consumption, timings, message sizes, etc.) can be found in [8] (Table 1).

Figure 4 shows the results of the simulation for one day. It depicts the energy consumed per reading considering a uniform backoff window between 0 and 5 (*highly synchronized*), 10, 30, and 60 seconds (*non-synchronized*). As the num-

ber of devices and the level of synchronization grow, the random-access opportunities (RAOs) per second grow as well, producing more and more collisions. These collisions cause retries, and a noticeable impact on energy consumption (up to 12 percent more energy per reading). Therefore, this use case shows the paramount importance of randomizing node activation in mIoT scenarios in order to avoid RAO peaks and premature battery drain.

Implementation Details: This scenario requires a single `meter` trajectory implementing the logic of each IoT device in an infinite loop, and *N* workers are attached to it at $t = 0$. Each device registers itself for a given signal ("reading"), and waits in sleep mode until a new reading is requested, which is triggered by a secondary trajectory (`trigger`). As soon as a new reading is signalled, the RA procedure starts by randomly selecting one of the 54 preambles available, which are defined as resources. The process of seizing a preamble encompasses two sub-trajectories:

- If there are no collisions, the preamble is successfully seized, and the `post.seize` sub-trajectory is executed, which transmits a reading.
- If there is a collision, rejection occurs, and the `reject` sub-trajectory is executed, which performs the RA backoff (for a random number of slots), and restarts the RA procedure (for a maximum of *m* retries).

Both sub-trajectories set the appropriate power levels *P* for the appropriate amount of time. In this case, these power levels throughout the simulation time are retrieved with the `get_mon_attributes()` method. Again, the energy is concisely computed, and represented using `dplyr` and `ggplot` packages.

Wrapup

Thanks to these scenarios, we have demonstrated the usability and suitability of `simmer` for fast prototyping of three different 5G scenarios. The code developed highlights some of the characteristics that make `simmer` attractive for researchers and practitioners in communications research:

- A novel and intuitive trajectory-based approach that simplifies the simulation of large networks of queues, including those with feedback.
- Flexible resources, with dynamic capacity and queue size, priority queueing and preemption.
- Flexible generators of arrivals that can draw inter-arrival times from any theoretical or empirical distribution via a function call.
- Asynchronous programming features and monitoring capabilities, which helps the researcher focus on the model design.

Table 1 summarizes the main simulation statistics for each scenario (simulation time is computed with a machine equipped with an Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz x4 (32 cores), and 64 GB of RAM, Debian GNU/Linux 8, R 3.3.2, and `simmer` 3.6). These numbers attest that `simmer` can be used to simulate relatively complex scenarios with very few lines of code (under 100 lines in all cases). Furthermore, the automatic monitoring capabilities embedded in `simmer`, and the integration with the R language, enable sophisticated analyses and visualizations with only a few more lines. It is likewise remarkable the ease with which multiple scenarios, with different parameters, can be simulated concurrently thanks to base R functions. Thus, exploring a large number of combinations of parameter values is not only straightforward, but also as fast as the slowest thread given a sufficient number of CPU cores available.

	Use case 1	Use case 2	Use case 3
Simulation time(s)	~ 10	~ 150	~ 150
No. of parallel scenarios	12	5	12
Max. events, 1 scenario	2 215 076	4 427 839	28 364 172
Total no. of events	14 565 424	11 196 337	98 952 165
Implementation lines	30	97	42
Analysis + plotting lines	28	18	14

Table 1. Overview of simulation features.

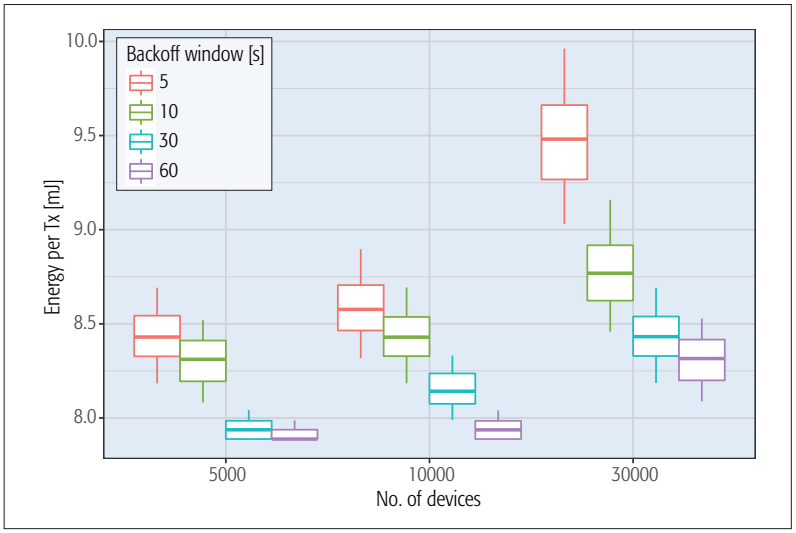

Figure 4. Energy consumption per transmission attempt for different traffic models and number of devices.

Conclusions

In this article, we have introduced the use of the `simmer` R package for the simulation of communication scenarios. We have illustrated its simple yet powerful syntax, and have demonstrated its ease of use and functionality with the analysis of three 5G-inspired scenarios, corresponding to radio, access, and metro deployments. The results obtained, which can be easily computed thanks to the powerful capabilities of R, help take design decisions related to hardware choices, traffic prioritization or access scheme configuration. Because of these, we believe `simmer` is a powerful tool to validate analytical studies, or to complement the use of more complex and costly simulation frameworks.

Acknowledgments

This article has been partially supported by the 5G-City project (TEC2016-76795-C6-3-R), and the TEXEO project (TEC2016-80339-R), both funded

It is likewise remarkable the ease with which multiple scenarios, with different parameters, can be simulated concurrently thanks to base R functions. Thus, exploring a large number of combinations of parameter values is not only straightforward, but also as fast as the slowest thread given enough number of CPU cores available.

by the Spanish Ministry of Economy and Competitiveness.

References

[1] I. Ucar, B. Smeets, and A. Azcorra, "`simmer`: Discrete-Event Simulation for R," *J. Statistical Software*, 2018.
[2] J. G. Andrews *et al.*, "What Will 5G Be?" *IEEE JSAC*, vol. 32, no. 6, June 2014, pp. 1065–82.
[3] A. Checko *et al.*, "Cloud RAN for Mobile Networks — A Technology Overview," *IEEE Commun. Surveys Tutorials*, vol. 17, no. 1, First Quarter 2015, pp. 405–26.
[4] A. D. L. Oliva *et al.*, "Xhaul: Toward an Integrated Fronthaul/Backhaul Architecture in 5G Networks," *IEEE Wireless Commun.*, vol. 22, no. 5, Oct. 2015, pp. 32–40.
[5] A. de la Oliva *et al.*, "An Overview of the CPRI Specification and its Application to C-RAN-based LTE Scenarios," *IEEE Commun. Mag.*, vol. 54, no. 2, Feb. 2016, pp. 152–59.
[6] G. Kramer, B. Mukherjee, and G. Pesavento, "IPACT: A Dynamic Protocol for an Ethernet PON (EPON)," *IEEE Commun. Mag.*, vol. 40, no. 2, Feb. 2002, pp. 74–80.
[7] C. Hoymann *et al.*, "LTE Release 14 Outlook," *IEEE Commun. Mag.*, vol. 54, no. 6, June 2016, pp. 44–49.
[8] P. Andres-Maldonado *et al.*, "Optimized LTE Data Transmission Procedures for IoT: Device Side Energy Consumption Analysis," *Proc. 2017 IEEE Int'l. Conf. Commun. Workshops* (ICC Workshops), May 2017, pp. 540–45.
[9] R. G. Cheng *et al.*, "RACH Collision Probability for Machine-Type Communications," *Proc. 2012 IEEE 75th Vehicular Technology Conf. (VTC Spring)*, May 2012, pp. 1–5.

Biographies

Iñaki Ucar (inaki.ucar@uc3m.es) received his M.Sc.Eng. in telecommunications engineering and the M.Sc. in communications from the Universidad Pública de Navarra (UPNA) in 2011 and 2013 respectively, and his M.Sc. in telematics engineering from the Universidad Carlos III de Madrid (UC3M) in 2014. Currently, he holds the position of teaching assistant and is pursuing his Ph.D. in the Department of Telematics Engineering at UC3M. His work focuses on energy efficiency of wireless networks.

José Alberto Hernández (jahgutie@it.uc3m.es) completed his five-year degree in telecommunications engineering at UC3M in 2002, and his Ph.D. degree in computer science at Loughborough University, Leicester, United Kingdom, in 2005. He has been a senior lecturer in the Department of Telematics Engineering since 2010, where he combines teaching and research in the areas of optical WDM networks, next-generation access networks, metro Ethernet, energy efficiency, and hybrid optical-wireless technologies. He has published more than 80 articles in both journals and conference proceedings on these topics. He is a co-author of the book *Probabilistic Modes for Computer Networks: Tools and Solved Problems*.

Pablo Serrano (pablo@it.uc3m.es) received his degree in telecommunications engineering and his Ph.D. from the Universidad Carlos III de Madrid (UC3M) in 2002 and 2006, respectively. He has been with the Telematics Department at UC3M since 2002, where he currently holds the position of associate professor. He has over 80 scientific papers in peer-reviewed international journal and conferences. He has served as a guest editor for *Computer Networks*, and on the TPC of a number of conferences and workshops including IEEE INFOCOM, IEEE WoWMoM and IEEE Globecom.

Arturo Azcorra (azcorra@it.uc3m.es) received his M.Sc. degree in telecommunications engineering from the Universidad Politécnica de Madrid in 1986 and his Ph.D. from the same university in 1989. In 1993, he obtained an M.B.A. from the Instituto de Empresa. He has a double appointment as a full professor (with chair) in the Telematics Engineering Department of the University Carlos III of Madrid, and as Director of IMDEA Networks. He has coordinated the CONTENT and E-NEXT European Networks of Excellence, has served as a program committee member in numerous international conferences, and has published over 100 scientific papers in books, international journals and conferences.