An open source platform for exploring NB-IoT system performance

Sergio Martiradonna, Alessandro Grassi, Giuseppe Piro, Luigi Alfredo Grieco, and Gennaro Boggia
Dept. of Electrical and Information Engineering, Politecnico di Bari, Bari, Italy
CNIT, Consorzio Nazionale Interuniversitario per le Telecomunicazioni
{name.surname}@poliba.it

***Abstract*—Applications and services emerged in the Internet of Things era are introducing new challenging requirements to the current communication technologies. Since a high number of devices are intentioned to sporadically send small data packets, the transmission speed goes at a secondary level of importance. Instead, more attention is devoted to coverage, computational complexity, and energy constraints. In this context, the 3rd Generation Partnership Project is complementing existing mobile systems with a novel technology, namely NarrowBand Internet of Things (NB-IoT), which is expected to be more suitable for Internet of Things scenarios. While it is getting more and more attention from academia and industry, there is an increasing need to have flexible instruments for designing and testing protocols and algorithms for the NB-IoT stack. To this end, the work discussed herein proposes an open source simulator for NB-IoT, conceived starting from the well-known LTE-Sim tool. At the present stage, it implements uplink transmission with both Single-Tone and Multi-Tone configurations, random access procedure, and baseline scheduling, while offering the support for multi carrier, stand alone, in band, and guard band operating modes. The tool has been used for evaluating the performance of reference NB-IoT scenarios. Obtained results highlight the impact of traffic load and system configurations on network performance.**

***Index Terms*—NB-IoT, simulation tool, performance evaluation**

I. Introduction

Thanks to the fast growth of the Internet of Things (IoT) phenomenon, several novel services are emerging in smart cities, logistics, industrial control, and many more application domains. Thus, an increasing number of smart devices joining the worldwide Internet are expected to be very small, transmit only small amounts of data, and be equipped with slim batteries able to supply energy for weeks, months, or even years of operation. While cutting-edge wireless technologies, such as Long Term Evolution (LTE) and LTE-Advanced (LTE-A), have impressively achieved outstanding performance, they cannot suitably work in typical IoT scenarios [1]. In fact, they were initially conceived for more traditional applications, such as web browsing and file transfer, where throughput and spectral efficiency are the key performance figures. Conversely, these requirements are not primarily targeted anymore in the IoT. Instead, the focus is on coverage, computational complexity, and energy constraints.

Since Release 13, the Third Generation Partnership Project (3GPP) provided a concrete answer to these challenging issues. In fact, it conceived a novel technology, namely NarrowBand IoT (NB-IoT), more suitable for IoT application domains [2]. NB-IoT natively offers the support for a massive number of devices, wide-area coverage, device complexity reduction, long battery lifetime, and flexible deployment. Compared to LTE, it leverages a narrower bandwidth, simpler modulation, and lower data rates. Therefore, as a matter of fact, NB-IoT can be classified as a new independent air interface, harmoniously integrated within the overall LTE network architecture.

As both academia and industry are increasingly involved in the development of NB-IoT, the need for suitable simulation tools, to be used for designing and optimizing new protocols and algorithms, as well as highlighting possible problems before that actual prototypes are built, increases as well. At the time of this writing, preliminary NB-IoT implementations were proposed for both ns-3 [3] and OPNET [4] network simulators. But, the former is largely incomplete, while the latter only focuses on the link level and is not freely available for the research community.

To bridge this gap, the work presented herein proposes an open-source implementation of the NB-IoT communication model, conceived starting from the well-known LTE-Sim tool [5]. Specifically, it implements many features, including uplink communications, random access procedure, a new scheduling engine running round-robin and first-in first-out strategies, and Single-Tone and Multi-Tone transmission configurations with different subcarrier spacing and number of tones. Moreover, it offers a basic support for multi carrier, stand alone, in band, and guard band operating modes. By maintaining the typical flexibility of LTE-Sim, it is implicitly open to future upgrades. To provide a further insight, preliminary simulations evaluated the performance of baseline IoT scenarios. Obtained results show that the random access procedure has the most significant impact on the system performance, and the difference between the available schedulers can only be observed by using sufficiently large packets.

The rest of the paper is organized as follows: Section II describes the main details of the NB-IoT standard, while Section III presents the developed simulation tool, and Section IV presents some preliminary results. Finally, Section V concludes the paper.

II. NB-IoT in a nutshell

The physical layer of NB-IoT is based on Orthogonal Frequency Division Multiplexing (OFDM) and requires a substantially smaller bandwidth than LTE, equal to 180 kHz for both downlink and uplink [6]. Multiple carriers can be used to provide more capacity. In this case, the *anchor carrier* conveys synchronization, broadcast information and paging messages. All the idle devices use this carrier. On the other hand, connected devices can be served on this anchor carrier as well as exchange data on a *non-anchor carrier*.

Three different operation modes are supported: stand alone (an operator can replace one GSM carrier of 200 kHz with NB-IoT, leaving a guard interval of 10 kHz on both sides of the channel), in-band (one or more NB-IoT channels can be deployed inside an LTE channel) and guard band (one or more channels within the guard-band of LTE bandwidth can be allocated to NB-IoT).

In downlink, NB-IoT fully inherits from LTE: a minimum burst of bits can be sent by the base station to the mobile user through the entire bandwidth of 180 kHz (divided in 12 subcarriers with a spacing Δf = 15 kHz) and over a Transmission Time Interval (TTI) equal to 1 ms. In uplink, instead, two main configurations are supported, that are *Single-Tone* and *Multi-Tone*. When the Single-Tone is used, the bandwidth of 180 kHz can be divided in either 48 or 12 subcarriers. In the first case, the system adopts a subcarrier spacing Δf = 3.75 kHz and each subcarrier can be assigned to a single user only. Accordingly, 48 users can be scheduled at the same time during a TTI equal to 32 ms (see Fig. 1). The second case uses a spacing Δf = 15 kHz and 12 different users can be scheduled over a 8 ms TTI. For the Multi-Tone configuration, the subcarrier spacing is set to 15 kHz. Moreover a number of 3, 6 or 12 subcarriers can be assigned to the single user. Depending on the selected configuration, the TTI and the number of users schedulable at the same time changes accordingly (see Fig. 1 for more details). Anyway, for both Single-Tone and Multi-Tone configurations, the Resource Unit (RU) is the smallest unit to map a transport block [7].

NB-IoT leverages the existing LTE physical channels, including Narrowband Physical Downlink Shared Channel (NPDSCH), Narrowband Physical Downlink Control Channel (NPDCCH) and Narrowband Physical Broadcast Channel (NPBCH) for the downlink and Narrowband Physical Uplink Shared Channel (NPUSCH) for the uplink. Additionally the Narrowband Physical Random Access Channel (NPRACH) is properly defined for NB-IoT for enabling the random access procedure [8]. NPRACH only uses the Single-Tone configuration, with Δf = 3.75 kHz, and provides pseudo-random frequency hopping between groups of symbols. There are four groups in the preamble, and each of them comprises five symbols plus the cyclic prefix. The resulting algorithm, offers different congestion free preambles as there are subcarriers allocated to the NPRACH. Also, up to three NPRACH resource configurations can be configured in a cell, each one corresponding to a different coverage level. In particular, a resource configuration is described through periodicity, number of repetitions, starting time, frequency location, and number of subcarriers. In NB-IoT the random access procedure is always contention based [7] and envisages the exchange of

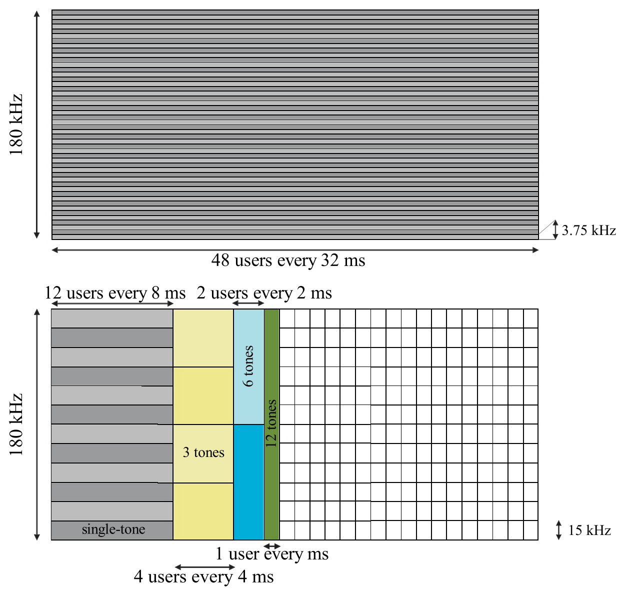


Fig. 1: Resource configuration in NB-IoT uplink

four messages. First, the mobile terminal transmits a random access preamble. Second, the base station transmits a Random Access Response that contains timing advance command and scheduling of uplink resources for the mobile terminal, to be used next. Third, the end user transmits its identity to the network using the scheduled resources. The third message can also convey the Data Volume Indicator, which informs the base station that the end user has data to transmit, thus triggering the consequent uplink scheduling. Finally, the base station transmits a contention-resolution message to resolve any contention which may be caused by multiple users transmitting the same random access preamble.

NPUSCH has two formats: Format 1 is used for carrying uplink data, Format 2 is used for signaling Hybrid Automatic Repeat reQuest (HARQ) acknowledgement for NPDSCH. The maximum number of repetitions of the NPUSCH transmission configured for the user depends on its coverage level. As shown in TABLE I, Transport Block Size (TBS) of NPUSCH Format 1 is rather low, regardless of the particular Modulation and Coding Scheme (MCS) chosen for the transmission. In particular, 11 and 14 MCS different indexes are supported when the Single-Tone and Multi-Tone configurations are used, respectively [9].

III. The implemented simulator

The proposed tool was developed starting from the well-known LTE-Sim simulator [5]. In most cases, the changes preserve the original code and maintain a high level of backwards compatibility. Fig. 2 shows a high level illustration of the main features implemented in the simulator. More technical details are presented below.

TABLE I: TBS for NPUSCH Format 1 (additional indexes for *Multi-Tone* are presented in gray)

I_{MCS}	RUs							
	1	2	3	4	5	6	8	10
0	16	32	56	88	120	152	208	256
1	24	56	88	144	176	208	256	344
2	32	72	144	176	208	256	328	424
3	40	104	176	208	256	328	440	568
4	56	120	208	256	328	408	552	680
5	72	144	224	328	424	504	680	872
6	88	176	256	392	504	600	808	1000
7	104	224	328	472	584	712	1000	1224
8	120	256	392	536	680	808	1096	1384
9	136	296	456	616	776	936	1256	1544
10	144	328	504	680	872	1000	1384	1736
11	176	376	584	776	1000	1192	1608	2024
12	208	440	680	1000	1128	1352	1800	2280
13	224	488	744	1032	1256	1544	2024	2536

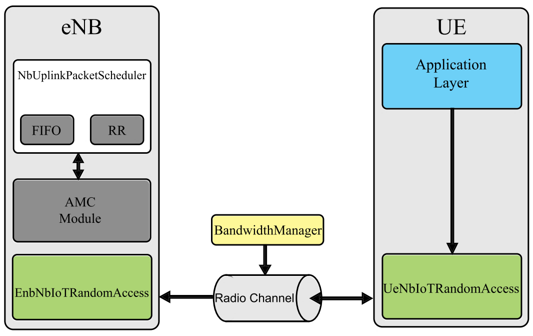


Fig. 2: Main functionalities of the implemented NB-IoT tool

A. Radio Resource Configuration

NB-IoT deeply changes radio resource configuration, especially with respect to the uplink side. Thus, the **BandwidthManager** class of LTE-Sim has been drastically changed to support both uplink subcarrier spacings, that are $\Delta f = 3.75$ and $\Delta f = 15$ kHz. The new implementation enables both Single-Tone and Multi-Tone transmissions (the latter can be configured for using either 3, 6 or 12 tones). Multiple carriers can be enabled by simply defining more channels of 180 kHz each to be used for data transmission. Moreover, by modifying the set of frequencies to use, the **BandwidthManager** class natively supports stand alone, in band, and guard band operating modes.

The TTI length of the uplink can be set via the **FrameManager::setTTIlength()** method. In essence, it gives the correct length of the TTI, based on the actual number of tones and the subcarrier spacing Δf chosen for the simulation.

The Adaptive Modulation and Coding module is used to choose the TBS, starting from the selected MCS index and number of RUs given by the scheduling strategy. To this end, a new class has been created, namely **nbAMCmodule**, encompassing the TBS retrieval procedure. To the best of authors' knowledge, the current scientific literature and reference 3GPP specifications do not provide a complete list of BLock Error

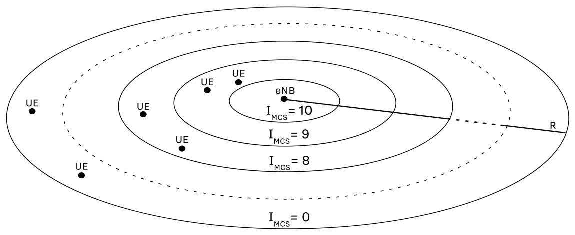


Fig. 3: Distribution of MCS indexes in a cell

Rate (BLER) curves for the NB-IoT, yet. Thus the Adaptive Modulation and Coding process is simplified as follows: the cell is divided into 11 or 14 equally wide concentric zones (see Fig. 3); each zone is denoted by a different MCS, decreasing from first zone to the last one (this way, different MCS indexes are distributed along the cell, becoming inversely proportional to the distance between mobile terminals and base station.

B. Random Access Procedure

The original version of LTE-Sim does not support the random access procedure. The developed code allows to control RACH opportunities occurrence, as long as number of different preambles. In case of a preamble collision, the procedure fails for all the involved devices. Then it can be repeated for a maximum number of times, which can be configured by the user.

Two completely new classes have been added, **UeNbIoTRandomAccess** and **EnbNbIoTRandomAccess**. As soon as the traffic generator creates a packet at the application layer, the **UeNbIoTRandomAccess::StartRaProcedure()** method initializes the procedure. As a first step, the **UeNbIoTRandomAccess::SendMessage1()** method sends the preamble to the base station, randomly chosen among all the available preambles, upon the first NPRACH resource occurrence. Then, **EnbNbIoTRandomAccess::CheckCollision()** scans all the preambles to find collisions and, if appropriate, it sends a Random Access Response to end users, through **EnbNbIoTRandomAccess::SendMessage2()**, or it will tell the mobile terminal to try to send another preamble in the next NPRACH occurrence. In this case **UeNbIoTRandomAccess::ReStartRaProcedure()** first checks that the maximum number of retry attempts has not been reached, and then repeats all the previous operations. On the contrary, if the number of attempts has been reached, the random access procedure is immediately stopped. However, if no collision occurs, then the mobile terminal will receive the Random Access Response and it will call **UeNbIoTRandomAccess::SendMessage3()** in a specific TTI, indicated in the second message itself. Upon receiving the third message, the base station automatically sends back to the end user the last message, through **EnbNbIoTRandomAccess::SendMessage4()**. This method also finalizes the whole random access procedure and makes the end user active and able to transmit data.

UeNbIoTRandomAccess class takes care of configuring NPRACH resources, which are given by periodicity, number of repetitions, starting time, frequency location, and number of subcarriers. The developed code allows to set the periodicity and the amount of subcarriers to use, but it does not currently have the option of configuring the other parameters. By calling **SetRachReservedSubChannels()** periodically, the base station actually allocates resources to NPRACH if they meet the resource configuration (i.e., the above criteria). This means that NPRACH resources must not be used by users for NPUSCH transmission, and, at the same time, they are the only ones where users can send preambles. Therefore, before starting one of these two operations, mobile terminals must check the current resource type. This verification is done by means of the **isRachOpportunity()** method belonging to the **UeNbIoTRandomAccess** class.

C. Packet schedulers

The abstract class **nbUplinkPacketScheduler** has been implemented for providing essential and common methods to all the scheduling strategies, which revolutionize the baseline functionalities of LTE-Sim schedulers. Then, two different scheduler classes have been developed, related to two well-known strategies: first-in first-out (FIFO) and round-robin (RR).

The **nbFifoUplinkPacketScheduler** class implements the FIFO algorithm. It checks, at the beginning of each subframe, if the current TTI is dedicated to NPRACH or not. In the affirmative case, the scheduling procedure is interrupted. Otherwise it checks whether there are available subcarriers, and finally it allocates them to users who need to be scheduled, by following the order of arrival. A dummy delay is introduced in order to lengthen the transmission interval, which is actually finalized at the physical layer only at the expiration of this delay. In this way it is possible to consider the longer duration of NB-IoT TTIs. **nbRoundRobinUplinkPacketScheduler** implements the RR algorithm. It is not very different from the previous class, since the performed operations are essentially the same. The differences with the **nbFifoUplinkPacketScheduler** class are basically two. First, the number of RUs to be assigned to each user, that is the **ruslice** variable, spans from 1 to 10 and it is independent from the amount of data the end user has to transmit. Of course, if the mobile terminal needs fewer RUs to complete the transmission, only the necessary ones are allocated to avoid wasting resources. Second, the allocation takes place cyclically: as soon as all the scheduled users are served, the allocation procedure resumes from the beginning of the queue. When a user finishes to transmit its data, it is removed from the queue.

IV. Preliminary performance evaluation

The developed tool has been used to fulfill a preliminary performance evaluation of NB-IoT. Simulations consider a plausible IoT scenario, where devices generate at the application layer packets of either 128 or 256 Bytes every 60 s. Motionless mobile terminals are uniformly distributed within all the MCS zones of the cell with a radius of 1 km. Conducted tests evaluated the impact of the average number of transmission requests per second, λ, and the packet size, p on network performance. Since transmission requests follow a discrete uniform distribution in the interval $[1, \Delta T_p]$, every ΔT_p seconds there are, on average, $\lambda \Delta T_p$ different mobile terminals that want to transmit a packet of size p. System performance with both Single-Tone and Multi-Tone transmissions has been evaluated when both FIFO and RR scheduling algorithms are taken into account. Finally, relevant parameters of the simulated environment are given in TABLE II.

System performance was evaluated in terms of number of end users involved in the random access procedure, the average number of users managed by the packet scheduler, average system goodput, and end-to-end packet delays. In order to increase the level of confidence of reported results, each simulation has been repeated 50 times.

TABLE II: Simulation Settings

Parameter	Value
Cell Radius - R	1 km
NB-IoT Bandwidth	180 kHz
Duration	150 s
NPRACH Periodicity	320 ms
Packet Size - p	128 Bytes, 256 Bytes
Packet Generation Interval - ΔT_p	60 s
Requests per second - λ	20, 40, 60
Transmission Type	Single-Tone, Multi-Tone
Subcarrier Spacing - Δf	3.75 kHz, 15 kHz
Number of Tones	3, 6, 12
Scheduling Algorithm	FIFO, RR

First of all, Fig. 4 shows both the number of users that triggers and finalizes the random access procedure. Reported results are independent from the transmission configuration and the scheduling algorithm. As expected, greater λ values lead to a higher number of collisions. In fact, since more mobile terminals try to send a preamble in the same NPRACH occurrence, the collision probability increases accordingly. For this reason, the percentage of users that finalizes the procedure decreases with the traffic load. Specifically, when λ is less than 60, approximately all the mobile terminals complete the aforementioned procedure, although they require more attempts. Instead, when $\lambda = 60$, the collision probability becomes so high that even more users access the NPRACH, but only few of them are able to complete the procedure and transmit data packets. It is worth noting that all the curves do not grow indefinitely because each user can repeat the random access procedure for a maximum number of times for a given packet.

To better understand what happens after the end of the random access procedure, the average number of users that are managed by the scheduler when the packet size p is set to 128 Bytes and 256 Bytes is reported in TABLE III and TABLE IV, respectively. When p = 128 Bytes, the average number of users that wait to transmit data over the radio interface is generally lower than the number of users that

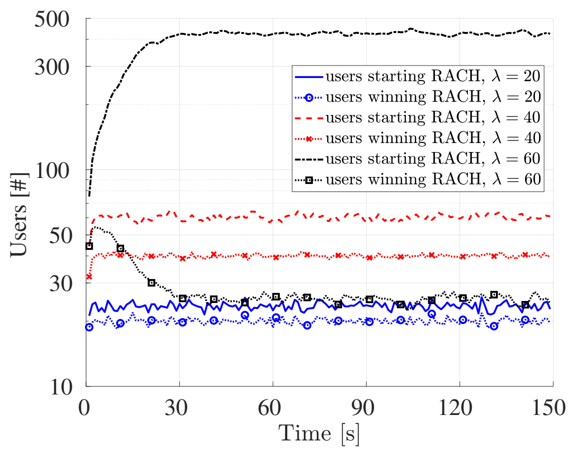


Fig. 4: Amount of users involved in the random access procedure

generates packets to transmit during the unit of time. This means that NB-IoT is able to manage the amount of requests that successfully passes the random access procedure. When p = 256 Bytes, instead, the amount of resources available in a simple configuration based on a 180 kHz carrier only is not suitable to support the overall traffic load. In this case, the system collapse and a very high number of users has to wait for a long time in the scheduling queue. In line with the results reported in Fig. 4, a significant reduction in the average number of schedulable users is registered when $\lambda = 60$. In fact, many users are unable to complete the random access procedure and transmit a packet. In general, both schedulers offer a similar behavior. But, in a scenario with $\lambda = 40$ and Multi-Tone configuration, the RR scheduler handles queued users in a more efficient manner.

Fig. 5 shows the average system goodput. When p = 128 Bytes, obtained results are nearly independent from the transmission type, Δf, number of tones, or the adopted scheduling algorithm. The results do not follow a monotonic behavior: moving from $\lambda = 40$ to $\lambda = 60$, the network registers a performance degradation. This is not caused by radio channel errors (which are not implemented yet), but it is due to the limited number of mobile terminals that completed the random access procedure (see Fig. 4). These considerations are still valid also in the case p = 256 Bytes. Nevertheless, Fig. 5b demonstrates how Single-Tone mode is indeed capable of handling a higher numbers of transmission request. In fact, more users can be scheduled at the same time during a TTI compared to the Multi-Tone configuration. The goodput is indeed improved.

Lastly, Fig. 6 shows the cumulative distribution functions of the end-to-end packet delays, calculated by considering the influence of random access procedure, scheduling decisions, and physical transmission. From one side, when p = 128 Bytes, both schedulers reach a similar performance. The reason is that the base station is able to empty the scheduling queue before the arrival of new requests. Therefore, scheduling decisions

TABLE III: Average number of scheduled users, p = 128 Bytes

Scheduling Algorithm	**Number of Tones**	λ			Δf
		20	40	60	
FIFO	1	13.5	26.8	23.9	3.75 kHz
		3.3	7.4	6.6	15 kHz
	3	1.8	6	6	
	6	1.5	5.5	5.5	
	12	1.4	5.3	5.1	
RR	1	13.3	26.9	23.6	3.75 kHz
		3.4	7.4	5.7	15 kHz
	3	1.8	5.9	5.4	
	6	1.5	5.5	5.3	
	12	1.4	5.3	5.2	

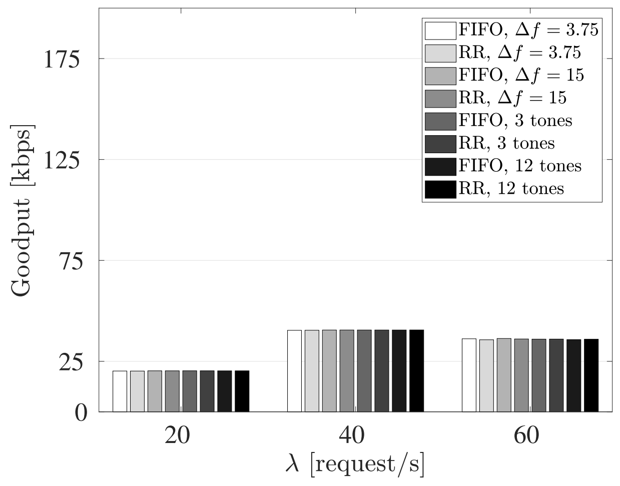
TABLE IV: Average number of scheduled users, p = 256 Bytes

Scheduling Algorithm	**Number of Tones**	λ			Δf
		20	40	60	
FIFO	1	25	235	106	3.75 kHz
		7	202	70	15 kHz
	3	5	794	252	
	6	4	796	251	
	12	5	807	263	
RR	1	25	277	106	3.75 kHz
		7	254	76	15 kHz
	3	5	771	288	
	6	5	772	272	
	12	5	762	299	

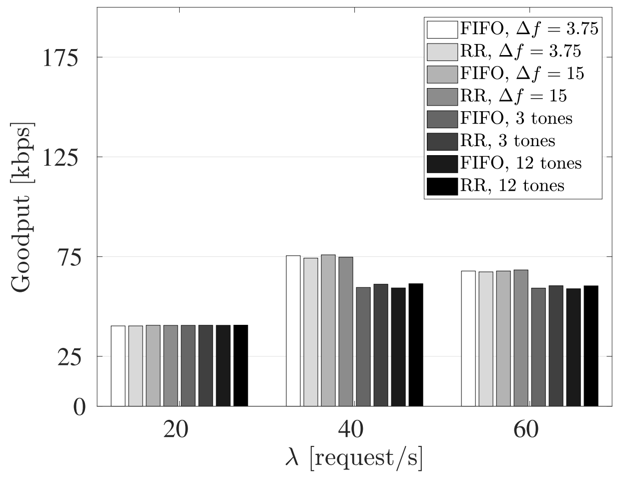
bring negligible differences. At the same time, however, packet delays grow with the value of λ. When λ increases, in fact, more users perform the random access procedure and the amount of time needed to complete it increases as well. Also, when $\Delta f = 3.75$ kHz (see Fig. 6a and Fig. 6b), longer duration of the TTI certainly gives rise to greater delays. But, a higher number of subcarriers translates into a limited susceptibility to the λ variation (the three curves are closer), since it allows simultaneous transmission of a greater number of users. On the other side, when p = 256 Bytes and λ is greater than 20, the difference between scheduling policies becomes more noticeable. In particular, RR guarantees lower delays with respect to FIFO for most of the users. Also, when $\lambda = 60$ higher delays are registered even if the number of users that are actually scheduled are less than the other cases (as confirmed by both Fig. 4, TABLE III, and TABLE IV). This solely depends on the longer time needed by mobile terminals to successfully complete the random access procedure.

V. Conclusion

This contribution presented the first open source simulator tool modeling the NB-IoT stack. Its main functionalities (which include random access procedure, scheduling algorithms, and radio resource configurations) are evaluated through baseline simulations. Moreover, obtained results confirm the expected behavior of implemented models. Future activities intend to extend the developed tool with additional features, like the support of simultaneous Single-Tone and Multi-Tone transmission schemes, repetitions, physical errors, more accurate channel and interference models for all the standardized NB-IoT operation modes.

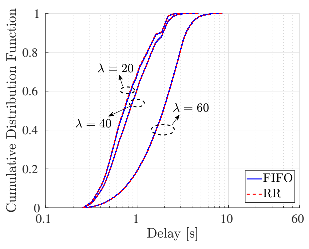


(a) p = 128 Bytes

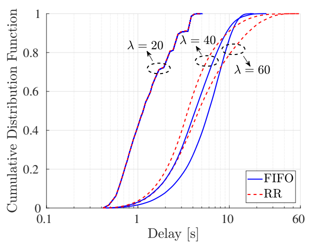


(b) p = 256 Bytes

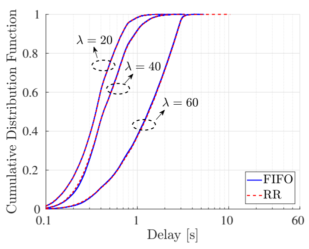
Fig. 5: Average Goodput



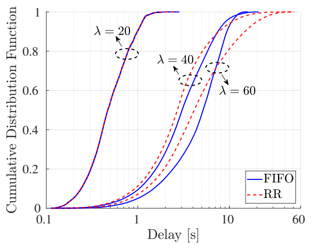
(a) Single-Tone $\Delta f = 3.75$ kHz p = 128 Bytes



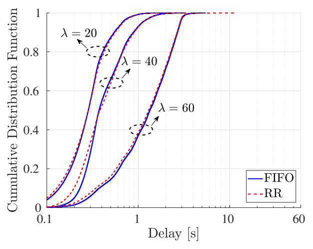
(b) Single-Tone $\Delta f = 3.75$ kHz p = 256 Bytes



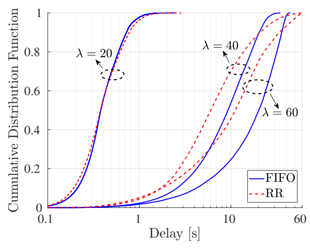
(c) Single-Tone $\Delta f = 15$ kHz p = 128 Bytes



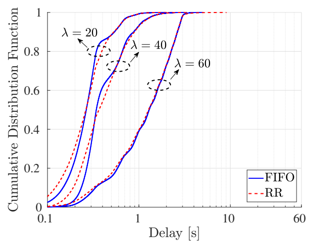
(d) Single-Tone $\Delta f = 15$ kHz p = 256 Bytes



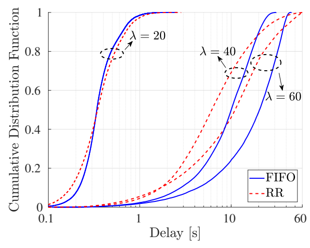
(e) 3 tones p = 128 Bytes



(f) 3 tones p = 256 Bytes



(g) 12 tones p = 128 Bytes



(h) 12 tones p = 256 Bytes

Fig. 6: Cumulative distribution functions of end-to-end packet delay

Acknowledgements

This work was framed in the context of the project SymbIoTe, which receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement 688156. It has also been partially funded by the FFABR fund (by the Italian MIUR).

References

[1] A. Laya, L. Alonso, and J. Alonso-Zarate, "Is the Random Access Channel of LTE and LTE-A Suitable for M2M Communications? A Survey of Alternatives." *IEEE Communications Surveys and Tutorials*, vol. 16, no. 1, pp. 4–16, 2014.

[2] R. Ratasuk and N. Mangalvedhe and Y. Zhang and M. Robert and J. P. Koskinen, "Overview of narrowband IoT in LTE Rel-13," in *Proc. of 2016 IEEE Conference on Standards for Communications and Networking (CSCN)*, Oct. 2016.

[3] S. Foni, T. Pecorella, R. Fantacci, C. Carlini, P. Obino, and M.-G. Di Benedetto, "Evaluation methodologies for the NB-IOT system: issues and ongoing efforts," in *Proc. of AEIT International Annual Conference*. IEEE, 2017.

[4] Y. Miao, W. Li, D. Tian, M. S. Hossain, and M. F. Alhamid, "Narrow Band Internet of Things: Simulation and Modelling," *IEEE Internet of Things Journal*, 2017.

[5] G. Piro, L. A. Grieco, G. Boggia, F. Capozzi, and P. Camarda, "Simulating LTE cellular systems: An open-source framework," *IEEE transactions on vehicular technology*, vol. 60, no. 2, pp. 498–513, 2011.

[6] 3GPP, "LTE physical layer; General description (Release 14)," TS 36.201, Mar. 2017.

[7] ——, "E-UTRA and E-UTRAN; Overall description (Release 14)," TS 36.300, Jun. 2017.

[8] X. Lin, A. Adhikary, and Y. P. E. Wang, "Random Access Preamble Design and Detection for 3GPP Narrowband IoT Systems," *IEEE Wireless Communications Letters*, vol. 5, no. 6, pp. 640–643, Dec 2016.

[9] 3GPP, "E-UTRA; Physical layer procedures (Release 14)," TS 36.213, Jun. 2017.