Everyone can slice LoRaWAN

Thibaut Bellanger, Alexandre Guitton, Razvan Stanica, Fabrice Valois

Everyone can slice LoRaWAN

Thibaut Bellanger*, Alexandre Guitton*†, Razvan Stanica*, Fabrice Valois*
*Univ Lyon, INSA Lyon, Inria, CITI, F-69621 Villeurbanne, France
†Université Clermont-Auvergne, CNRS, Mines de Saint-Étienne, Clermont-Auvergne-INP, LIMOS, F-63000 Clermont-Fd

***Abstract*—Long-Range Wide Area Networks (LoRaWAN) enable low-power data collection over long distances, and they are thus widely used for Internet of Things applications, despite their limitations to meet some traffic requirements (*e.g.*, reliability). To achieve the quality of service required by the applications, service differentiation is a promising approach which can be provided by network slicing. In this work, we show that related works are either incompatible with the LoRaWAN specifications, or do not isolate traffic, or assume an a priori known and stable traffic. In this paper, we propose a lightweight approach to achieve slicing in LoRaWAN, compatible with the LoRaWAN specifications. Our approach allows to isolate traffic, to protect confirmed traffic, and to deal with unsolicited traffic.**

I. Introduction

Long-Range Wide Area Network (LoRaWAN) is a wireless communications standard dedicated to data collection from low-power end-devices over long distances. It is widely used in Internet of Things (IoT) applications for three main reasons: (i) it can trade-off communication range with data rate, (ii) it enables the deployment of low-cost networks, and (iii) it is simple to implement. Applications using LoRaWAN typically include smart city monitoring, smart farming, and supply chain management and optimization.

Because of the simple design of its medium access control (MAC) protocol, LoRaWAN is not able to support the Quality of Service (QoS) requirements of some IoT applications. For this reason, several proposals ([1]–[3]) were made to use slicing in order to implement service differentiation over LoRaWAN, i.e., to provide different levels of QoS. The idea of slicing comes from 5G networks. It uses independent resources, including radio resources, to create slices which are isolated from each other, then allocates nodes to slices, and manages these slices independently. In theory, a slicing approach applied to LoRaWAN could provide different QoS levels for various types of traffic: confirmed, unsolicited, traffic from mobile end-devices, etc.

However, current slicing approaches for LoRaWAN fail to meet key requirements: (i) several approaches are not entirely compatible with the LoRaWAN specifications from the LoRa Alliance, (ii) most approaches do not isolate traffic, which does not allow to provide QoS guarantees, (iii) they require an a priori knowledge of the traffic, and assume stable traffic conditions, and (iv) they often assume that traffic parameters are controlled by the IoT node, which is not practical for real operated networks. Moreover, the current slicing approaches for LoRaWAN are unable to deal with unsolicited traffic, for instance coming from end-devices associated to an application server that has been terminated: such a situation is unfortunately likely to happen increasingly in the future, as some old application servers will eventually be switched off without removing the corresponding IoT nodes.

In this paper, we propose a lightweight slicing approach for LoRaWAN that: (i) is compatible with the LoRaWAN specifications, (ii) isolates traffic, (iii) is dynamic, and (iv) is controlled by the network-server. Our approach is flexible, and it can handle slices based on radio channels, spreading factors or other radio/network parameters. We show that we can take benefit from the adaptive data rate (ADR) mechanism to create slices and to allocate end-devices to slices. In other words, with our approach, *everyone can slice LoRaWAN*.

The remainder of this paper is organized as follows. Section II gives a brief overview of LoRaWAN and of slicing in 5G networks. Section III presents the related works on slicing for LoRaWAN, as well as on the ADR mechanism. Section IV introduces our lightweight approach to perform ADR-based slicing. Section V evaluates our approach through simulations. Finally, Section VI concludes our work.

II. Background

A. LoRaWAN network architecture and medium access

LoRaWAN defines the network architecture and MAC protocol of low-power wide area networks. It is based on the Long Range (LoRa) physical layer modulation. In LoRaWAN, end-devices send LoRa frames, which are received by (possibly multiple) gateways. The gateways forward them to a centralized network server, which deduplicates them. The network server then determines the target application from the headers, and forwards the frames to the corresponding application server. Both the application server and the network server can also send frames to end-devices: in this case, the network server decides through which gateway the frame should be sent. Note that a join procedure is required before an end-device can communicate.

LoRaWAN defines three classes of end-devices. We focus here on Class A, as it is the only mandatory class, and is used by a large majority of applications. In Class A, the end-devices use an Aloha MAC mechanism: they can transmit a frame without waiting or assessing the channel, but have to comply with a duty-cycle. Each transmission is followed by the opening of two short reception windows, where end-devices can receive acknowledgments in the case the frame they transmitted belongs to the confirmed traffic class.

The LoRa modulation integrates three main parameters: the spreading factor (SF), the channel bandwidth (BW) and the

coding rate (CR). These three parameters have a direct impact on the robustness of the modulation: a long communication range can be reached with a small BW, a large SF and a large CR. LoRaWAN can dynamically adapt both BW and SF through a mechanism called ADR. When ADR is active, the network server constantly monitors the received signal strength of frames reaching the gateways, for each end-device. Then, it chooses LoRa parameters for each end-device. Finally, the network server uses a control message to request the end-device to switch to this set of parameters.

Typical LoRaWAN applications produce traffic with different QoS requirements: (i) the traffic might be confirmed (that is, each uplink frame is requesting for an acknowledgment) or unconfirmed, (ii) the end-devices can be mobile (in which case, the ADR has to be disabled and end-devices use SF12) or static (in which case, the ADR is usually enabled), and (iii) there might be unsolicited end-devices producing traffic, typically when the application server has been switched off while some end-devices are still deployed and running.

The simple Aloha mechanism used in LoRaWAN does not allow for service differentiation to provide QoS guarantees. However, LoRaWAN uses several orthogonal frequency channels and orthogonal SF at the physical layer. These two orthogonalities are used by most slicing approaches, as we will see later.

B. Network slicing

Slicing is a key feature in 5G networks, allowing the support of a wide range of applications with very different QoS requirements. This is made possible by the virtualization of all resources (*e.g.*, radio, backhaul, computation) and of all the network functions in the cellular network. The main idea is to use orthogonal (or independent) resources and independent functions to design slices based on the application QoS requirements. Each slice is isolated from the others, monitored and managed independently, and can evolve dynamically according to the network demand. Because of this isolation, there is no impact of the traffic from a given slice on another one, which allows to enforce the key performance indicators.

To provide slicing in a network architecture, it is mandatory to identify independent resources. Then, a slice is defined as the list of the necessary virtual network functions, the dedicated radio resources, and the required network hardware.

III. Related works

There are only a few existing slicing approaches for LoRaWAN. In this section, we compare them based on three crucial properties: the isolation guarantees, the support of dynamic traffic, and the compatibility with the LoRaWAN specifications (especially on the end-device side). We also give a brief overview of proposed ADR enhancements.

A. Slicing in LoRaWAN

In [1], the authors propose a three-step approach to perform slicing in LoRaWAN, while assuming an a priori knowledge of the traffic. First, they group end-devices with similar QoS requirements, which determines the slices. Second, they compute the capacity for each slice and allocate channels accordingly. Third, each end-device is managed by the best gateway. The authors study the performance of their approach according to various SF allocation strategies for the third step.

In [4], the authors propose to change the SF allocation based on the device priority. They propose two greedy algorithms that allocate the SF to high-priority end-devices first: one algorithm allocates the smaller SFs first, while the other allocates the larger SFs first. However, the approach assumes that all end-devices can use the smallest SF, which is not realistic in a real deployment.

In [5] and [6], the authors propose two similar approaches to provide different QoS, expressed as differentiated targets of frame delivery rate. They propose to separate the traffic based on frequency channels, and to further limit the number of devices per SF. The authors also propose to reduce the duty-cycle of low-priority nodes in order to limit the traffic. In both papers, some end-devices are forbidden to send traffic, which requires to modify the behavior of end-devices, and thus does not follow the LoRaWAN specifications.

In [2], the authors explain that slicing on LoRaWAN channels does not provide a fine QoS granularity, as the number of channels in LoRaWAN is typically very small (from 3 to 8 in the European regional settings). Thus, they propose end-devices of different priorities to be allocated to the same channel. However, they introduce a blocking probability on end-devices, in order to reduce the traffic of low-priority ones. This proposal requires to modify the end-devices to support priorities and a blocking probability, which is not supported by the LoRaWAN specifications.

In [3], the authors propose an approach using Deep Deterministic Policy Gradient to optimize resource allocation within frequency-based slices. For each gateway, an agent attempts to find the optimal pair of SF and transmission power for each node. To reduce the learning time, the authors use Transfer Learning. However, the proposal has two main drawbacks. First, there is no isolation since the slices are defined by SF and transmission power. Second, the authors assume that each gateway can allocate channels to end-devices independently, while the broadcast nature of LoRa communications generates interference when two neighboring end-devices communicate on the same channel for different gateways.

In [7], the authors define three classes of traffic, with reliability and urgency components. Those three classes are isolated in frequency-based slices. The authors then use Deep-Q Networks to allocate efficiently SF and transmission power to each node. Each slice uses a different reward function which enables to define different allocations to ensure QoS requirements. However, the authors achieve slightly better scores than the ADR allocation scheme.

Table I summarizes these approaches by considering the three properties which appear imperative for providing slices in LoRaWAN: the ability to isolate slices, the ability to deal with dynamic traffic without an a priori knowledge, and the

Table I: Summary and comparison of related works.

Reference	Slicing approach	Isolation	Dyamic traffic support	LoRaWAN Specification compatibility
[1]	frequency-based	+	-	+
[4]	SF-based	+	-	+
[5]	SF-based and frequency-based	+	-	-
[6]	SF-based and frequency-based	+	-	-
[2]	frequency-based	-	-	-
[3]	frequency-based and power-based	-	-	+
[7]	frequency-based	+	+	+
Our approach	SF-based and frequency-based	+	+	+

compatibility with the LoRaWAN specifications.

B. ADR modifications

The ADR mechanism enables the network server to allocate the SF and transmission power to each end-device, based on the quality of the links with gateways. However, several major weaknesses were identified in this mechanism. First, the ADR allocates parameters only based on the received signal strength, without taking into account the other end-devices. This causes congestion in the network, typically when too many end-devices are allocated to the same SF. Second, the ADR is unstable when channel conditions are volatile or for mobile end-devices. Consequently, many researchers have proposed modifications to the ADR mechanism, typically to improve the throughput, to reduce the energy consumption, and to increase capacity. More details on these ADR enhancements can be found in [8].

IV. ADR-BASED SLICING FOR LORAWAN

A. Motivations

We aim to design a slicing architecture for LoRaWAN, while remaining compatible with the LoRaWAN specifications. In other words, our proposal should be supported by any real LoRaWAN deployment, without requiring any modification on either the end-devices, the join server or the application servers. We propose to leverage the ADR mechanism to allocate and manage the slices, in addition to its usual role of improving the link budget of LoRa devices.

The goal of using slices is to protect the traffic through isolation in order to meet the QoS requirements of the applications. In the context of LoRaWAN, we claim that the confirmed traffic suffers from a lack of reliability in practice. Thus, we propose to create a slice dedicated to end-devices sending confirmed traffic. Moreover, in LoRaWAN networks, we also observe unsolicited traffic, *i.e.* traffic from end-devices whose application server is stopped. Indeed, when an application server is switched off, thousands of end-devices may continue to send traffic on their own, which drastically impacts the performance of the whole network. Thus, we propose to dedicate a slice to isolate such unsolicited traffic. Finally, we dedicate a third slice to the end-devices which send only non-confirmed traffic, which we refer to as normal traffic. Note, that we can extend these three slices to a larger number of slices, *e.g.*, a slice to isolate mobile end-devices.

B. How to slice LoRaWAN?

According to the design principles of slices in 5G, we should slice a LoRaWAN network both in the back-end network (network server, application servers, join server), and in the radio access part (end-devices, gateways).

1) The back-end network side: Because of the low traffic intensity of IoT applications, originating from small IoT packet sizes (typically in the order of tens of bytes), we claim that slicing the back-end is not necessary. If it is requested by some applications, for privacy issues or performance reasons, a virtual local area network can easily isolate the traffic, satisfying these QoS constraints.

2) The radio access side: Slicing the radio resource part of a LoRaWAN network can be done either by using LoRa frequency channels, or by using LoRa spreading factors. Without loss of generality, we propose to consider a slicing approach based on the use of spreading factors. Our work can be easily extended to frequency channel slicing, or to both frequency channel and SF slicing.

We chose to use SF to achieve slicing for two key reasons:

- *Quasi-orthogonality*: SF are often claimed to be orthogonal and independent. In practice, the SF are quasi-orthogonal to one another [9]. Table II summarizes the impact of each SF on the others. If chosen correctly, a specific SF can be used to protect the confirmed traffic, a second one to isolate unsolicited traffic, and all the others for the non-confirmed traffic.
- *Allocation and reallocation*: LoRaWAN typically uses the ADR mechanism to optimize the link budget through a dynamic adaptation of the SF used by the end-devices. Without modifying the end-device behavior, we extend the ADR to also change the SF according to the slicing strategy. Based on the header of LoRa frames, the network server is able to detect devices sending mostly confirmed traffic. Based on the `application_id` of each frame, it is also able to determine if the corresponding application server is running: if not, we assume that the traffic is unsolicited.

Table II: Interference matrix of Spreading Factor [9].

	SF7	**SF8**	**SF9**	**SF10**	**SF11**	**SF12**
SF7 victim	76%	5%	1%	0%	0%	0%
SF8 victim	17%	80%	1%	0%	0%	0%
SF9 victim	17%	6%	80%	0%	0%	0%
SF10 victim	18%	6%	2%	80%	0%	0%
SF11 victim	18%	6%	2%	0%	81%	0%
SF12 victim	19%	6%	2%	0%	0%	64%

C. ADR-based slicing for LoRaWAN

1) Description of the slices: Our proposal supports the following three slices: (i) the *Normal slice* manages all the traffic coming from end-devices which do not have strong reliability requirements, and whose application server is still running, (ii) the *Confirmed slice* is dedicated to support only the confirmed traffic, *i.e.*, the traffic which requires to be acknowledged, (iii) the *Unsolicited slice* is a slice which receives all the unsolicited traffic, coming from end-devices which are still running despite having an inactive application server. Note that these latter nodes produce unwanted traffic.

2) Frames management and slice allocation: Each LoRa frame is received by gateways, then forwarded to the network server (NS). The NS analyzes the header of each LoRa frame to determine the corresponding slice. For this, we use information already existing in the LoRaWAN messages. If the message type in the MAC header of the LoRa frame is set to `100`, the traffic is confirmed. During a join procedure, which allows an end-device to be associated to a LoRaWAN, the `application_id` is set. We use this field to keep track of end-devices sending data to application servers that are not reachable, and thus to identify unsolicited traffic. The traffic generated by the other nodes is classified as normal. In each case, the NS uses our ADR-based slice mechanism to allocate the corresponding SF to the end-devices.

3) How to slice using SF?: We investigate two strategies to create the three slices:

- Lower SF Slicing (LSF-Slicing): We can see from Table II that SF7 is the SF being the least impacted by other SFs, whereas SF11 is the most impacted by the other SFs. For this reason, in this strategy, we use SF7 only for confirmed traffic, and SF11 only for unsolicited traffic. The remaining SFs are shared among normal end-devices, and are allocated using the usual ADR algorithm to improve the link budget.
- Higher SF Slicing (HSF-Slicing): SF12 is known to provide the better coverage and the higher robustness, despite its lower data-rate. We also recall that SF7 provides the lower time-on-air value, at the cost of a very limited radio range. Moreover, SF7 leads to a lower impact on the other traffic and generates low interference. For these reasons, in this strategy, we use SF12 only for confirmed traffic, and SF7 only for unsolicited traffic. The remaining SFs are shared among normal end-devices, and are again allocated using the usual ADR algorithm.

Note that normal end-devices cannot use either SF7/SF11 (LSF-Slicing case), or SF7/SF12 (HSF-Slicing case). In the event that the legacy ADR algorithm would recommend one of these forbidden SFs to a normal device, we allocate a larger SF if available, or a smaller SF otherwise.

Table III: Summary of SF allocation per slice (Normal, **C**onfirmed, **U**nsolicited), for the two strategies.

Strategy	SF7	SF8	SF9	SF10	SF11	SF12
LSF-Slicing	***C***	*N*	*N*	*N*	***U***	*N*
HSF-Slicing	***U***	*N*	*N*	*N*	*N*	***C***

V. Performance evaluation

We used the FLoRa module [10] for the OMNeT++ simulator [11] to run our simulations. We deployed either 100 or 250 end-devices in a square area of 300 m$\times$300 m. The single gateway is deployed at the center of this area, in a square of 50 m$\times$50 m. Traffic is generated using a Poisson process, where the inter-arrival rate follows an exponential distribution of intensity 1/600 (including for the first frame). We randomly selected either 5% or 30% of the end-devices to produce confirmed traffic, and either 5% or 30% of the end-devices to produce unsolicited traffic. We used a single frequency channel with a duty-cycle of 1%, in order to focus on the impact of the SF. All the other end-devices produce normal traffic. Each simulation lasts for eight hours. Each scenario is repeated ten times, with random positions for the end-devices and gateway.

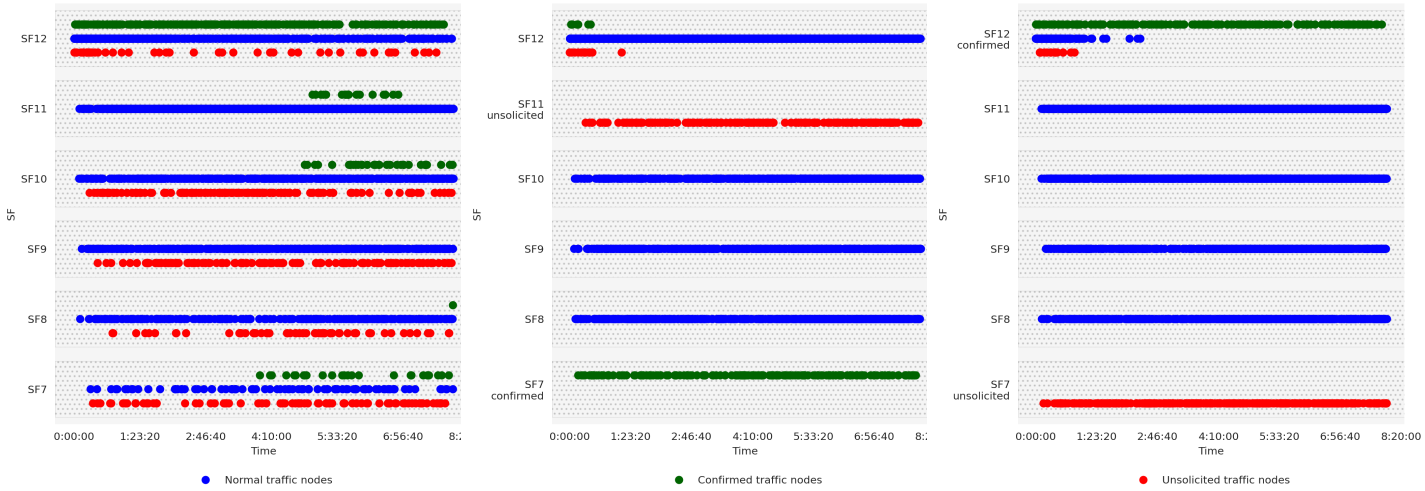
A. Slices can be isolated

Figure 1 shows the distribution of the SF as a function of time, for the baseline LoRaWAN (on the left), the LSF-Slicing strategy (on the middle) and the HSF-Slicing strategy (on the right), on a single instance of 100 nodes with 5% confirmed traffic and 5% unsolicited traffic. As expected, the baseline LoRaWAN allocates the SF independently of the type of traffic. However, both our strategies can isolate the traffic in the correct slices, as soon as the join procedure has been finished. Thus, **our approach is able to isolate slices using independent resources**. The allocation of the correct slice requires the end-device to receive a message from the network server. However, this requires some time, as downlink opportunities are scarce in LoRa and are heavily influenced by the duty-cycle limitation (1% in Europe). We call this delay the *slicing completion delay*, and we define it as the longest time required by a node to be allocated to its correct slice, once it has initiated the join procedure. We can see from Figures 1b and 1c that the slicing completion delay is lower than 3 hours, which is reasonable. Note that the slice allocation does not require the network server to collect 20 frames of history, unlike the usual ADR, but only a single one.

B. The confirmed traffic is protected

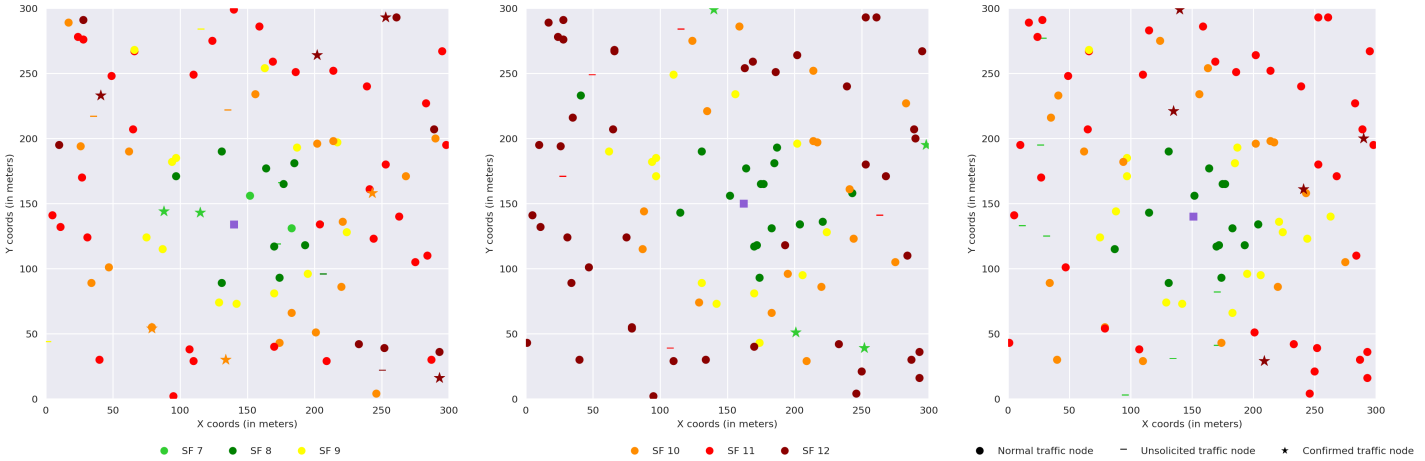
Figure 2 shows the spatial distribution of the SF, for all strategies, on a single instance of 100 nodes with 5% confirmed traffic and 5% unsolicited traffic. The baseline ADR of LoRaWAN allocates SF according to the received power at the gateway, that is based on an approximation of the distance. The LSF-Slicing strategy allocates SF7 (in light green) to end-devices generating confirmed traffic (represented with stars), and SF11 (in red) to end-devices generating unsolicited traffic (represented with a hyphen). The other SFs are allocated based on an approximation of the distance. Similarly, the HSF-Slicing strategy allocates SF12 (in brown) to confirmed end-devices (with stars), and SF7 (in light green) to unsolicited end-devices (with hyphen).

Figure 3 shows the packet delivery ratio (PDR) per type of traffic and per strategy, for three configurations: 100 nodes,



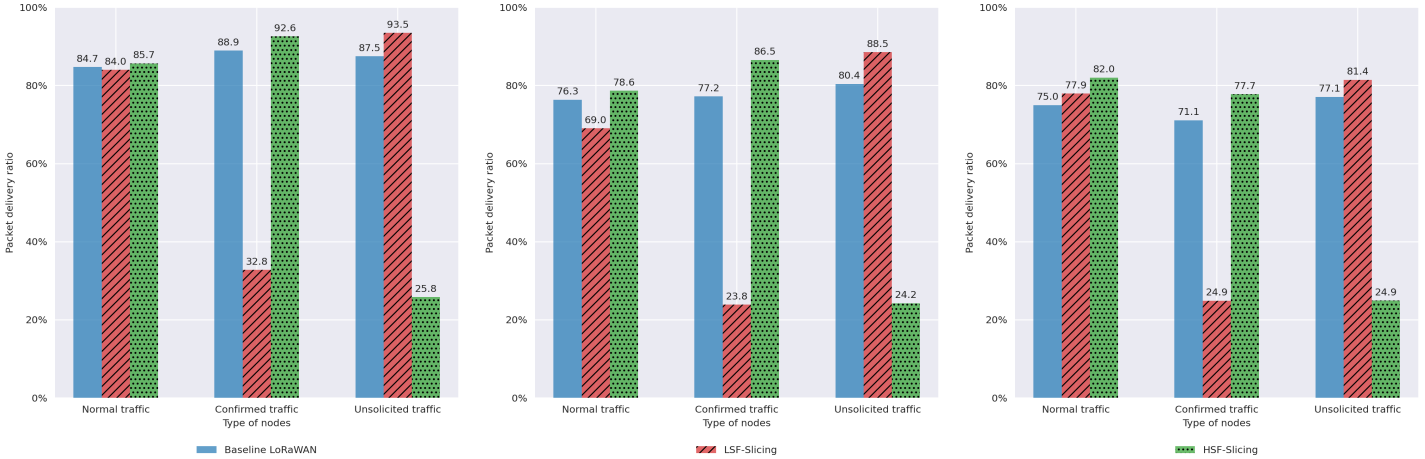
(a) Baseline LoRaWAN (b) LSF-Slicing (c) HSF-Slicing

Figure 1: Usage of SF as a function of time, on a single instance of 100 nodes, 5% confirmed traffic and 5% unsolicited traffic.



(a) Baseline LoRaWAN (b) LSF-Slicing (c) HSF-Slicing

Figure 2: Spatial distribution of SF, on a single instance of 100 nodes, 5% confirmed traffic and 5% unsolicited traffic.

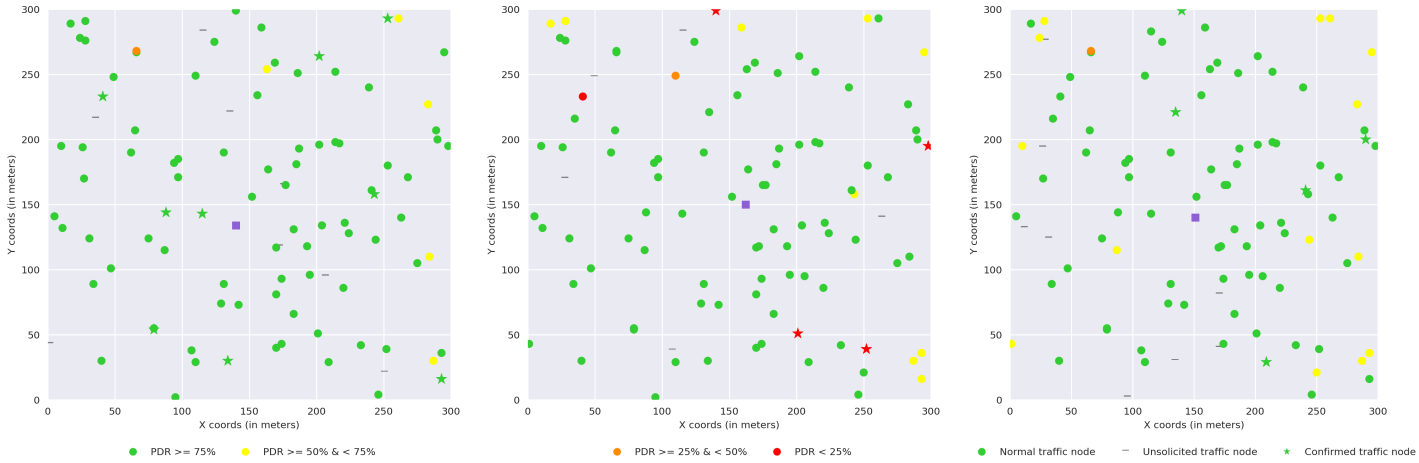


(a) 100 nodes, 5% confirmed traffic, 5% unsolicited traffic

(b) 250 nodes, 5% confirmed traffic, 5% unsolicited traffic

(c) 250 nodes, 30% confirmed traffic, 30% unsolicited traffic

Figure 3: Packet Delivery Ratio per type of traffic and per strategy, for three configurations.



(a) Baseline LoRaWAN (b) LSF-Slicing (c) HSF-Slicing

Figure 4: Spatial distribution of PDR, on a single instance of 100 nodes, 5% confirmed traffic and 5% unsolicited traffic.

5% confirmed traffic and 5% unsolicited traffic (on the left), 250 nodes, 5% confirmed traffic and 5% unsolicited traffic (on the middle), and 250 nodes, 30% confirmed traffic and 30% unsolicited traffic (on the right). The LSF-Slicing strategy achieves a very low performance for the confirmed traffic, in all configurations. Indeed, while SF7 enables high throughput, end-devices producing confirmed traffic are not necessarily close to the gateway, and thus might not be able to communicate with it using SF7. **The HSF-Slicing strategy is able to improve the PDR of the confirmed traffic** compared to the baseline LoRaWAN, for two reasons: it uses a robust SF for this traffic, while ensuring that the number of end-devices using this SF is not too large. **The HSF-Slicing strategy also improves the PDR of the normal traffic**, by completely removing the congestion due to unsolicited traffic. Finally, it yields a low PDR for unsolicited traffic, which is not an issue.

Figure 4 shows the spatial distribution of the PDR for the three strategies, on a single instance of 100 nodes, 5% confirmed traffic and 5% unsolicited traffic. With the LSF-Slicing strategy, it can be seen that a low PDR is experienced by confirmed end-devices that are not located close to the gateway. With the HSF-Slicing strategy, a low PDR is experienced by normal end-devices that are far away from the gateway. Those normal end-devices would have been allocated to SF12 with the baseline ADR, but SF12 is not available for normal end-devices in HSF-Slicing.

C. Limitations of our approach

The main limitation of our approach is the identification of end-devices sending unsolicited traffic. In our case, we considered only one gateway, connected to a NS who is aware of the state of all application servers (active or not). In practice, some application servers might be reachable only through some specific NS associated to their gateways. Therefore, not all NS have knowledge about all application servers, meaning that our classification solution for unsolicited traffic might generate false positives. A way to solve this issue would be to consider a federated approach in which application designers register their end-devices with an expiration date.

VI. Conclusion and Perspectives

In this paper, we proposed a lightweight slicing architecture which is fully compatible with the LoRaWAN specifications. We have shown that it is possible to exploit the ADR mechanism in order to manage and to allocate slices. We have shown that the HSF-Slicing strategy is able to perform traffic differentiation with a small convergence time, and it can improve the performance of both the confirmed traffic and the normal traffic.

In the future, we will consider end-devices joining and leaving the network at any time, as well as mobile end-devices. We will also add a slice monitor in order to dynamically adapt the size of the slices to the number of end-devices they serve.

References

[1] S. Dawaliby, A. Bradai, and Y. Pousset, "Adaptive dynamic network slicing in LoRa networks," *Future Generation Computer Systems*, vol. 98, p. 697–707, Sep 2019.

[2] G. Dandachi and Y. Hadjadj-Aoul, "A frequency-based intelligent slicing in LoRaWAN with Admission Control Aspects," in *ACM MSWiM*, Montreal, Québec, Canada, 2022.

[3] T. Mai, H. Yao, N. Zhang, W. He, D. Guo, and M. Guizani, "Transfer reinforcement learning aided distributed network slicing optimization in industrial IoT," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 6, pp. 4308–4316, 2022.

[4] S. Aggarwal and A. Nasipuri, "QoS based spreading factor assignment for LoRaWAN networks in IoT applications," in *SoutheastCon*, Mobile, AL, USA, 2022.

[5] A. Aimi, F. Guillemin, S. Rovedakis, and S. Secci, "Packet delivery ratio guarantees for differentiated LoRaWAN services," in *GLOBECOM*, Rio de Janeiro, Brazil, 2022.

[6] ——, "Traffic Control and Channel Assignment for Quality Differentiation in Dense Urban LoRaWANs," in *ACM WiOpt*, Turin, Italy, 2022.

[7] A. Tellache, A. Mekrache, A. Bradai, R. Boussaha, and Y. Pousset, "Deep reinforcement learning based resource allocation in dense sliced LoRaWAN networks," in *IEEE ICCE*, Virtual, 2022.

[8] R. Kufakunesu, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on adaptive data rate optimization in LoRaWAN: Recent solutions and major challenges," *MDPI Sensors*, vol. 20, no. 18, 2020.

[9] O. Seller, "LoRaWAN link layer," *Journal of ICT Standardization*, vol. 21, 2021.

[10] M. Slabicki, G. Premsankar, and M. Di Francesco, "Adaptive configuration of LoRa networks for dense IoT deployments," in *IEEE/IFIP NOMS*. Taipei, Taiwan: IEEE, 2018.

[11] "Omnet++ discrete event simulator." [Online]. Available: https://omnetpp.org/