Dynamic Network Slicing for LoRaWAN

Samir Dawaliby, Abbas Bradai, Yannis Pousset
XLIM Laboratory
University of Poitiers
Poitiers, France
samir.dawaliby@univ-poitiers.fr, abbas.bradai@univ-poitiers.fr, yannis.pousset@univ-poitiers.fr

Roberto Riggio
FBK
CREATE-NET
Trento, Italy
rriggio@fbk.eu

***Abstract*—One of the most important novelty in 5G is network slicing proposed as a collection of logical network functions for services running on a common physical device. In an Internet of Things (IoT) context, network resources need to be efficiently reserved and assigned for IoT devices in an isolated manner to handle and support specific Quality of Service (QoS) requirements for each slice. The focus of this paper is to investigate network slicing in LoRa network and propose a dynamic inter-slicing algorithm based on a maximum likelihood estimation that avoids resource starvation and prioritizes a slice over another depending on its QoS requirements. Moreover, we place the emphasis on a novel intra-slicing strategy that maximizes resource allocation efficiency of LoRa slices with regard to their delay requirements. After integrating an energy module for LoRa in NS3, simulation results performed in realistic LoRa scenarios highlight the utility of our dynamic network slicing proposition in providing isolation between slices with specific QoS guarantees.**

***Index Terms*—Internet of Things (IoT), LoRa, resource allocation, dynamic, network slicing**

I. INTRODUCTION

By 2020, it is expected that the 5th generation (5G) wireless mobile networks will provide the means to allow an all-connected world of objects and humans offering with it the needed flexibility to manage heterogeneous networks with major arising technologies namely software defined networking (SDN) and network functions virtualization (NFV). With the development of the latter, network slicing technology exhibits great potentials and is proposed to provide a feasible solution for 5G services [1]. 5G provision three services with conflicting quality of service (QoS) requirements (i.e., enhanced mobile broadband (eMBB), ultra-reliable low-latency communications (URLLC) and massive machine-type communications (mMTC)). Machine type communication devices (MTCDs) require heterogeneous behavior mainly when it comes to reliability and latency requirements for delay-sensitive devices. Therefore, network slicing came out as potential solution that allows multiple virtual networks with various QoS to be created on top of a common physical substrate, being mutually instantiated on-demand with independent management.

The massive number of IoT devices continuously increasing and connecting alongside mobile devices to the new generation core network (NGCN) in 5G, brings an exceptional need for bigger flexibility on network slicing definition and virtualization. This leads to new challenges in designing a dynamic network slicing and resource allocation strategy, which must guarantee the slicing isolation principle and simultaneously facilitates for the infrastructure provider resource allocation for different architectures deployed in a cost-effective manner.

Recently, few research works tackled network slicing for the internet of things, with focus on machine critical communications over various wireless networks. The work in [2] introduce a slicing infrastructure for 5G mobile networking and summarize research efforts to enable end-to-end network slicing between 5G use cases: eMBB, mMTC and URLLC. Authors in [3] focused on URLLC and proposed several slicing methods for URLLC scenarios which require strong reliability and latency guarantees. Furthermore, authors in [4] and [5] adopted network slicing in LTE mobile wireless networks. The former propose a dynamic resource reservation for machine-to-machine (M2M) communications whereas the latter present a slice optimizer component with a common objective in both papers to improve QoS experience of devices mainly reliability and delay requirements. In a 5G wearable network, authors in [6] took advantage of slicing technology to enhance the network resource sharing and energy-efficient utilization. Moreover, authors in [7] perform slicing in virtual wireless sensor networks to improve lease management of physical resources with multiple concurrent application providers.

At this stage, guaranteeing service-based QoS requirements in LoRa wireless access network (LoRaWAN) with traffic slicing remain as open research issues [8]. Therefore, unlike the previous work, we aim in this article to investigate network slicing in LoRa technology which, to the best of our knowledge, has not been treated before by the research community. LoRa proved to date to be one of the most promising wireless technologies for IoT because it meets low data rates requirements and offers the flexibility needed between throughput and range [9]. The main contribution is to extend the latter with a dynamic network inter-slicing algorithm based on a maximum likelihood estimation and an intra-slicing algorithm that meets the QoS requirements of each LoRa device depending on its running application. The remainder of this paper is organized as follows. We devote Section II to present the system model and a description of the problem is established in which we present the latter's constraints. Section III presents the proposed slicing algorithm implemented over the LoRa module of NS3 simulator [10]. We evaluate the performance of the algorithm and analyze simulation results carried out through various scenarios in Section V. Finally, Section VI concludes the paper.

978-3-903176-14-0

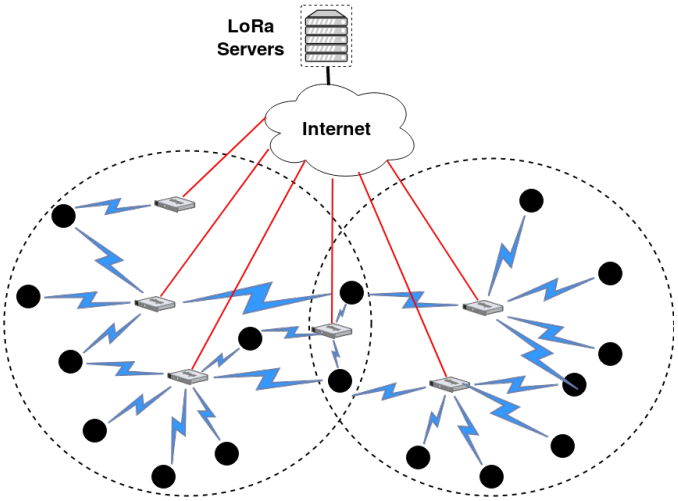


Fig. 1. LoRa-like architecture for IoT

II. System Model and Problem Formulation

A. System Model

As illustrated in **Fig. 1**, we consider a LoRa-like network for IoT, consisting of a set of $K = \{1, 2, ..., k\}$ MTCDs and $M = \{1, 2, ..., m\}$ LoRa Gateways (GWs) randomly distributed over a cell and connected to the cloud LoRa Servers via fronthaul links. Compared to other technologies in the IoT market such as Sigfox [11] and NB-IoT [12], LoRa is more resilient to jamming and interference [9] due to its ability to trade communication range against high data-rates in an efficient manner.

Network slicing mainly brings flexibility to the network by virtually reserving physical resources in order to meet the QoS requirements of each slice. We define a slicing framework that consists of a set of L virtual network slices such that $L = \{1, 2, ..., l\}$ can be created on physical network hardwares (i.e., GW, server, etc.). In IoT, MTCDs are imposed by a set of QoS requirements depending on the running IoT application that may require critical delay deadlines. The main goal behind slicing is to virtually split the network by reserving resources for each slice on the same physical device with each slice l characterized by a priority sp_l and a bandwidth $b_{l,m}$ at the GW level. We define a set of virtual flows F where a device k associated to slice l generates a flow $f_{k,l,m}$ that goes from the GW m to LoRa servers and is characterized by a utility metric $U_{k,l,m}$ which will be specified later on in this paper. The gateways in range will receive the packets but only one slice in a gateway will forward the packet to LoRa servers to avoid duplicated packets. LoRa is based on LoRaWAN protocol which uses chirp spread-spectrum (CSS) modulation at the physical layer described in the IEEE standard 802.15.4 [13]. Regarding channel modeling, CSS modulation transmits symbols by encoding them into multiple signals of increasing or decreasing radio frequencies making signals more robust to multi-path interference, Doppler shifts and fading [14]. Each MTCD k experiences a signal-to-interference-plus-noise ratio (SINR) from each GW m that depends on the spreading factor (SF) adopted by each device for information transmission. LoRa spreads each symbol in a rate of 2^{SF} chips per symbol with $SF = \{7, ..., 12\}$ resulting a data rate computed as written in **Eq. 1** below:

$$R_{k,l,m} = SF.\frac{R_c}{2^{SF}} = SF.\frac{b_{l,m}}{2^{SF}} \quad bits/s \tag{1}$$

where R_c denotes the chip rate and $R_{k,l,m}$ the data rate achieved by a device k depending on the bandwidth assigned to slice l of LoRa gateway m. Increasing the spreading factor reduces the transmitted data rate, offers a better sensitivity and increases the strength of the signal at the receiver following to the **Eq. 2** below:

$$P^{rx}_{k,l,m} = \frac{P^{tx}_{k,l,m} g^{rx}_{k,l,m} g^{tx}_{k,l,m}}{L} e^{\xi} \tag{2}$$

where $P^{rx}_{k,l,m}$ and $P^{tx}_{k,l,m}$ denotes the received and transmitted power with a channel antenna gain expressed with $g^{rx}_{k,l,m}$ and $g^{tx}_{k,l,m}$ respectively. L is the path loss which depends on the distance between the transmitter and the receiver and e^{ξ} is the lognormal shadowing component with $\xi \sim N(0, \sigma^2)$. Regarding interference, we follow the assumptions in [10] where a packet should survive interference that comes from other LoRa transmissions. Each LoRa device experiences a SINR value computed based on the **Eq. 3** below:

$$SINR_{i,j} = \frac{P^{rx}_i}{\sigma^2 + \sum_{n \in \partial_j} P^{rx}_n} \tag{3}$$

where P^{rx}_i is the power of the packet n under consideration sent by device with $SF = i$ and ∂_j a set of interfering packets with a common $SF = j$. Hence, packet survives interference with all interfering packets if, summing the received power values for each SF, it satisfies a higher receiving power margin value than a threshold $V_{i,j}$ with $i, j \in \{7, ..., 12\}$ listed in the matrix below [15]. Each $V_{i,j}$ element in the matrix represent the co-channel rejection when considering all couples of SF, i.e. the power margin (dB) threshold of the signal that a packet sent with $SF = i$ must at least have in order to be successfully decoded over every interfering packet with $SF = j$.

$$\begin{Vmatrix} & SF_7 & SF_8 & SF_9 & SF_{10} & SF_{11} & SF_{12} \\ SF_7 & -6 & 16 & 18 & 19 & 19 & 20 \\ SF_8 & 24 & -6 & 20 & 22 & 22 & 22 \\ SF_9 & 27 & 27 & -6 & 23 & 25 & 25 \\ SF_{10} & 30 & 30 & 30 & -6 & 26 & 28 \\ SF_{11} & 33 & 33 & 33 & 33 & -6 & 29 \\ SF_{12} & 36 & 36 & 36 & 36 & 36 & -6 \end{Vmatrix}$$

In IoT dense environment, the propagation loss model is computed based on the log-distance propagation model following to the **Eq. 4** below:

$$L = L_0 + 10.n.log_{10}(\frac{d}{d_0}) \tag{4}$$

where L denotes the path Loss (dB), d the length of the path (m), n represents the path loss distance exponent, d_0 the reference distance (m) and L_0 the path loss at reference distance (dB). **Table I** summarizes the key denotations adopted in the paper.

B. Problem formulation

In this paper, network slicing optimization in IoT is threefold and involves: *1) MTCDs admission and association to slices*; *2) Dynamic flexible inter-slicing resources reservation*; *3) Intra-slice resources allocation*. In the first step, we search to define L slices based on the delay urgency factor then we assign each device to the slice that meets its service latency requirement. It is noteworthy that in IoT, the delay urgency factor represents the key index to define the priority of a device over another without neglecting the service type, packet loss rate and the load that results from the large amount of IoT devices simultaneously connected to the network. In the second step, we estimate the capacity needed c_l for each slice l based on which physical receive paths on each GW m will be reserved and characterized by a specific utility value $U_{k,l,m}$. Finally in the third step, we search to optimize intra-slice resource allocation by assigning each device in slice l to the most efficient virtual flow with the highest utility metric. Let $\alpha_{k,l} \in 0, 1$ be a binary variable that indicates whether a device k is associated with a flow $f_{k,l,m} \in F$. The goal is to maximize the number of LoRa devices assigned to virtual flows in a way that maximizes the utility function adopted by each slice members. Therefore, the slicing and resource allocation problem for IoT can be formulated as

$$Max \sum_{k \in K} \sum_{l \in L} \alpha_{k,l} U_{k,l,m}, \forall m \in M \tag{5}$$

subject to

$$C1 : \sum_{l \in L} \alpha_{k,l} = 1, \forall k \in K \tag{6a}$$

$$C2 : \sum_{k \epsilon K} \beta_{k,m} p_{k,l,m} \leq P_m^{max}, \forall m \in M, \forall l \in L \tag{6b}$$

$$C3 : \sum_{k \epsilon K} \alpha_{k,l} \beta_{k,m} r_{k,l,m} \leq R_{l,m}^{max}, \forall l \in L, \forall m \in M \tag{6c}$$

$$C4 : \beta_{k,m} \in \{0, 1\}, \forall k \in K, \forall m \in M \tag{6d}$$

where constraint (6a) ensures that each MTCD is associated to exactly one slice regardless of the chosen flows. The total transmission power of each GW m is limited in constraint (6b). Moreover, constraint (6c) guarantees the sum of uplink traffic sent by slice members do not exceed the maximum data rate capacity of the slice that can be sent through each gateway. Constraint (6d) ensures binary-association values $\beta_{k,m}$ of device k to GW m.

TABLE I
LIST OF PARAMETERS

Parameter	Parameter Name
M	the set of LoRa Gateways
K	the set of MTCDs
L	the set of Slices
K_l	the set of devices associated to slice l
∂_j	the set of packets with $SF = j$
$f_{k,l,m}$	virtual flow for device k in slice l through GW m
$U_{k,l,m}$	utility for device k in slice l on GW m
sp_l	slice priority of slice l
$b_{l,m}$	bandwidth assigned for slice l over GW m
P_m^{max}	maximum transmission power of GW m
$g_{k,l,m}$	power gain between a GW m and a MTCD k
e^{ξ}	lognormal shadowing component
$\alpha_{k,l}$	admission index of MTCD k to slice l
$\beta_{k,m}$	association index of MTCD k to GW m
u_k	urgency factor for MTCD k
d_k	instant packet delay for MTCD k
PDB_k	packet delay budget for MTCD k

III. The Proposed Slicing Method

In a centralized LoRa network, the general control plane and resource management module are moved to a management and control entity (MCE) in the cloud to ensure an efficient coordination of resources. Hence, LoRa servers will be the final decision maker of assigning the devices to the appropriate slice and defining the gateway that will transmit the packet following to a three-steps optimization algorithm. In the first step, each MTCD will be assigned to the slice that meets its QoS requirements based on a balanced iterative reducing and clustering method using hierarchies (BIRCH). Then, in the second step GW resources will be dynamically reserved for each slice based on a maximum likelihood estimation (MLE) before choosing the best gateway that will forward the packet to the server based on a maximum-utility algorithm.

A. BIRCH-based Slicing Definition

Due to the ultra-dense nature in an IoT, we adopt BIRCH algorithm [16] which belongs to the agglomerative hierarchical clustering family and was proven as the best available clustering method for handling large datasets [17] [18]. The main goal behind this method is to define slices by checking the QoS requirements of each MTCD and moving from a large set of MTCDs into a group of subsets with similar QoS requirements. We consider that MTCDs are dynamically joining or leaving a slice following to their QoS requirements. This process is controlled by the server in which it will associate a device to a network slice at each slicing interval.

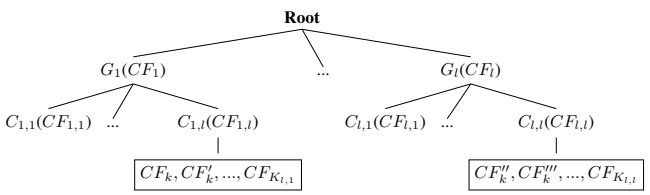


Fig. 2. BIRCH-based Slicing Tree

The most urgent MTCDs are the ones that have the closest instant delay d_k to their packet delay budget PDB_k and are assigned the highest priority. We denote u_k as the urgency factor of device k with $u_k = d_k/PDB_k$. Given K_l MTCDs in a cluster l, the latter will be considered as a utility point u_k of each MTCD device in a cluster with $k = 1, 2, ..., K_l$. Each node in the CF-tree is a cluster of subclusters defined by a clustering feature CF as follows:

$$CF = (K_l, LS, SS) = (K_l, \sum_{k=1}^{K_l} u_k, \sum_{k=1}^{K_l} u_k^2) \tag{7}$$

where K_l denotes the number of MTCDs in the cluster, LS the linear sum of the K_l utility points and SS the square sum of the K_l utility points. BIRCH dynamically builds a CF-tree, plotted in **Fig. 2**, at each time a new MTCD is inserted based on two parameters: a branching factor B and a threshold T.

Pseudo-code 1 BIRCH-based MTCDs-admission Algorithm

Input : Set of MTCDs K, diameter D, branching factor L, threshold T

```
1  begin
2      Initialize as many clusters as MTCDs
       for each k ∈ K do
3          Start from root
           Search for closest child node according to D
           Search for closest subcluster according to D
           if number of entries < T then
4              Add k to subcluster C_{l,l}; Update CF of C_{l,l}
5          else if number of childs < B then
6              Create a new subcluster C_{l,l'}
               Add k to C_{l,l'} ; Update CF of the parent node S_l
7          else if number of parents < B then
8              Split child nodes
               Redistribute CF entries according to closest D
9          else
10             Split parent nodes
11         end
12     end
13     Update CF entries in CF-tree
14 end
```

Output: Set of MTCD groups G_l(l=1,2,...,L)

Each parent node contains a maximum number of B childs and a single child node contains at most T entries. In this problem, B represents the number of L slices created with K_l the group of MTCDs admitted to slice l. Hence, l nodes derive from the root representing the slices created with each slice is made up of a group of MTCDs subclusters. In **Pseudo-code 1**, the algorithm scans the clusters from the root **(line 3)** and recursively traverses down the CF-tree and chooses the closest node at each level with the smallest average inter-cluster distance D as follows:

$$minD = \left(\frac{\sum_{k=1}^{K_l} \sum_{k'=K_l+1}^{K_l+K_{l'}} (u_k - u_{k'})}{K_l K_{l'}} \right)^{1/2}, \forall k \in K_l, \forall k' \in K_{l'} \tag{8}$$

After defining the candidate child node, a test is performed to find the closest CF-entry and define if the MTCD can be added to the candidate subcluster without violating the threshold condition. If so, group the node with the chosen entry and update the CF-entry of the candidate subcluster **(line 4)**. If not, a new entry is created for the node inside the candidate child node without breaking the branching factor condition **(line 5-6)**. Otherwise, we split the child node and redistribute the utility points based on the closest distance criteria to obtain a set of new subclusters that do not break the branching factor constraint **(line 7-8)**. In case the number of childs already reached the maximum, we split the parent nodes and the childs are redistributed to the closest parents **(line 9-10)**. We finally update all CF informations of the path from the inserted information to the root **(line 13)**.

B. Dynamic MLE-based Inter-Slicing Algorithm

Knowing that the physical capacity c of each GW's radio resources is limited. The goal here is to estimate and reserve the appropriate resources by finding the maximum likelihood buffer demands for each slice l starting by the one with the highest slicing priority. We assume that the traffic that needs to be uploaded follows a Poisson distribution and the servers are aware of the buffer status B_i of MTCDs in each slice l.

Lemma 1: Let T_i be the throughput needed by each MTCD $i, \forall i \in K_l$ captured at each slicing interval and identified by a corresponding probability distribution. For a fixed capacity, the optimum slicing strategy is to virtually reserve resources for each slice based on the mean throughput of its members.

Proof: We consider T_i follows a Poisson distribution P(λ_i) where λ_i denotes the throughput needed by device i assigned to slice $l, \forall i \in K_l$. Let $f(T_i|\lambda_i)$ be a probability density function similar to $L(\lambda_i|T_i)$ that represents the likelihood of λ_i given the observed throughput.

$$L(\lambda|T_1, T_2, ..., T_{K_l}) = f(T_1|\lambda_1) f(T_2|\lambda_2) .... f(T_l|\lambda_l)$$

$$L(\lambda|T_1, T_2, ..., T_{K_l}) = \prod_{i=1}^{K_l} \frac{e^{-\lambda_i} \lambda_i^{T_i}}{T_i!}$$

$$logL(\lambda|T_1, T_2, ..., T_{K_l}) = log\left[\prod_{i=1}^{K_l} \frac{e^{-\lambda_i} \lambda_i^{T_i}}{T_i!} \right]$$

$$logL(\lambda|T_1, T_2, ..., T_{K_l}) = \sum_{i=1}^{K_l} log\left[\frac{e^{-\lambda_i} \lambda_i^{T_i}}{T_i!} \right]$$

$$logL(\lambda|T_1, T_2, ..., T_{K_l}) = \sum_{i=1}^{K_l} \big[-\lambda + T_i log\lambda - log(T_i!) \big]$$

$$logL(\lambda|T_1, T_2, ..., T_{K_l}) = K_l \lambda_i + \sum_{i=1}^{K_l} T_i log\lambda_i$$

To find the maximum likelihood parameter, we apply the first derivative and solve it to zero

$$\frac{\partial logL(\lambda|T_1,T_2,...,T_{K_l})}{\partial\lambda} = -K_l + \frac{\sum_{i=1}^{K_l} T_i}{\lambda_i} = 0$$

$$\widehat{\lambda_i} = \frac{\sum_{i=1}^{K_l} T_i}{K_l}, \forall i \in K_l$$

Hence, $\widehat{\lambda_i}$ represents the optimal parameter estimation which proves that the optimal slicing decision is to consider the mean throughput of each slice members. However, slices are not equal in terms of priority. Therefore, resource on GWs will be dynamically allocated to the most urgent slice starting by the channel with the highest reliability. Let $\Theta_i = \widehat{\lambda_i} / \sum_{i=1}^{l} T_i$ be the slicing rate based on which the algorithm reserves for each slice a capacity $c_{i,m} = c_m.\Theta_i, \forall i \in L$. The algorithm starts with the most urgent slice, computes its needed throughput estimation $\widehat{\lambda_i}$ and defines Θ_i for channels reservation. The algorithm moves to the next slice and repeats the process until it serves all the slices or stops when no resources are left for the lowest priority slice.

C. Intra-Slicing Resource Allocation Algorithm

After defining and reserving the radio resources for each slice, we search in this section to maximize the utility function of MTCDs in each slice. Here, utility function for each slice is computed based on multiple criteria weights w_r and w_{ld} for reliability and load respectively manipulated using the analytical and hierarchy process (AHP) approach. The latter is proved as a verydecent method for multi-criteria decisions and was adopted by a large number of applications [19]. Based on the QoS table proposed in **Table II**, one can note that in IoT, MTCDs can be classified into three categories:

1) High critical communications (HCC): require the highest priority due to urgent delay communications and reliability, i.e: surveillance, emergency alerting and alarm monitoring. Based on **Eq. 9**, U_{HCC} is computed to define the utility for critical communications MTCDs with $\sigma_r = SINR_{k,l,m}/SINR_{max}$ the rate of reliability of SINR that a device k achieves on a flow $f_{k,l,m}$ over the highest flow reliability that can be achieved through slice l and δ_r, a binary variable that guarantees a minimum threshold when searching for the highest reliability links.

$$U_{HCC} = \delta_r(\sigma_r w_r) \quad with \quad \delta_r \in \{0,1\} \tag{9}$$

2) Medium critical communications (MCC): require lower priority consideration and are less critical in terms of delay. This slice presents a trade-off between reliability and load, i.e: health sensors and home security systems.

$$U_{MCC} = \sigma_r w_r + \sigma_{ld} w_{ld} \tag{10}$$

TABLE II
APPLICATION PARAMETERS [20]

QCI	SLice ID	Packet Delay Budget	Services	Percentage of IoT flows
5	1	<100 ms	Surveillance and Emergency Alerting	10 %
1-2	2	100-1000 ms	Health Sensors	15 %
3-4	2	100-1000 ms	Home Security System	15 %
6	3	>1000 ms	Smart Metering Applications	60 %

3) Low critical communications (LCC): require the lowest priority due to their non-guaranteed data rate and delay-tolerant QoS requirements, i.e: smart metering applications.

$$U_{LCC} = \sigma_{ld} w_{ld} \tag{11}$$

The algorithm searches in each slice for the gateway with the most robust and reliable link that offers the lowest delay [21], finds the highest U_{HCC} metric and allocates resources accordingly. Adding load to U_{HCC} metric could play a negative role in finding the most reliable link for a small number of urgent devices. However, unlike HCC slice, for a greated number of devices in U_{MCC} and U_{LCC} slices, load is important because in some cases the most reliable link may be overloaded due to the massive number of devices and should not be taken into consideration. Hence in **Eq. 10**, U_{MCC} is defined to search for the optimal flow that searches for a trade-off solution that offers the highest reliability with the minimum load possible. And finally in **Eq. 11**, LCC slice includes delay-tolerant devices with high packet delay budgets which justify U_{LCC} metric calculation in a load-aware manner without taking reliability into consideration. In **Fig. 3**, we consider a directed network $N = (V, E)$, where each MTCD k is a source node s uploading traffic to the internet considered as a sink node t such that $s, t \in V$. Moreover, each GW m is considered as edge node and bounded by the amount of flow allowed for MTCDs in each slice l. For each MTCD k in l, we search in the network for the flow that maximizes its utility function. Without loss of generality, we assume that no edges enter the sources or exist sinks. For each edge, we consider the respective utilities $U'_{k,l,m}$ and $U''_{k,l,m}$ in the network based on **Eq. 12** below:

$$U_{k,l,m} = U'_{k,l,m} + U''_{k,l,m} \tag{12}$$

Each MTCD k assigned to slice l search for the most efficient virtual flow with the objective to find the highest utility metric $U_{k,l,m}$ following to **Pseudo-code 2** below.

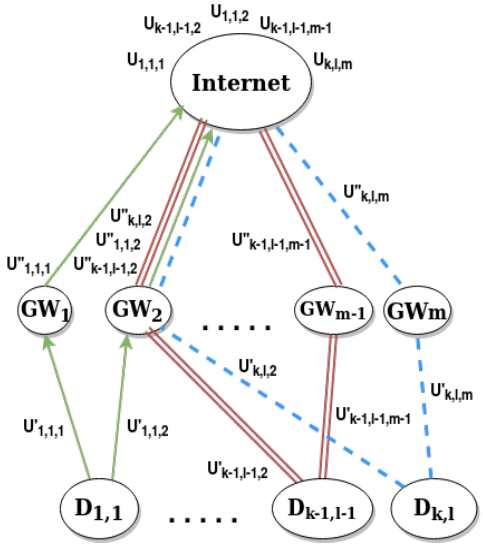


Fig. 3. Flow modeling for IoT Network Slicing

Pseudo-code 2 Max-Utility Inter-Slice Resource Allocation

Input : Set of MTCDs K, GWs M, slices L and capacity c

```
begin
    for each GW i ∈ M do
        Put slices in increasing order based on sp_l
        Reserve c_{l,i} based on MLE estimation
        Initialize flow utilities to null for all e ∈ E
    end
    for each slice l ∈ L do
        for each MTCD k ∈ K_l do
            Draw network N(V, E)
            Find path with the highest utility U_{k,l,m}
            Allocate device k to f_{k,l,m}
            Update capacity c_{l,m}
        end
    end
end
```

Output: Max-Utility flows allocation for MTCDs

IV. Simulations and Results

In uplink, centralized LoRa servers enable the opportunity to make efficient slicing configurations based on data traffic in MTCD's buffer. In this work, we used the LoRa model implemented by [10] in the open source NS3 simulator [22]. Simulations are replicated 50 times with respect to the parameters shown in the first section of **Table III**. We realize the experiment in a realistic LoRa scenario where MTCDs are randomly uploading small packet data to LoRa servers but in a periodic manner. While fixing the number of LoRa gateways to 4, the number of devices is increased till it reaches 1000 devices in a single cell. Simulations start with 100 devices to emulate a load of one due to the legal duty-cycle limitations of 1% in the European region [23]. LoRa devices and gateways are both placed over a cell of 7.5 KM radius following to a uniform random distribution. Each device is configured with spreading factors that varies from 7-12 when uploading traffic to LoRa GWs characterized by 8 receive paths in the 867-868 MHz european sub-band. Based on the **Eq. 13** below, we seek to evaluate the energy consumed by each MTCD when we increase the number of LoRa devices in each slice.

$$E_{k,l,m} = \frac{p_i^{tx} + p_i^{rx}}{V + epa}.d_{tx/rx} \quad (13)$$

where $E_{k,l,m}$ is the energy consumed by an IoT device, V the LoRa supply voltage, epa the amplifier's added efficiency, d_{tx} the duration of transmission, p_i^{rx} the power of reception and p_i^{tx} the power of transmission that vary between 2 and 14 dBm based on the spreading factor i adopted. However, energy is not implemented yet for the LoRa module. Therefore, we added an energy module inspired by the one that already exists for Wifi module and we configured the energy parameters following to the power model adopted for LoRa in [14] and [24] as shown in the second section of **Table III** below.

A. Energy Consumption

When increasing the number of nodes, the total energy consumed increases for all the simulated slices, as plotted in **Fig. 4** below. However, HCC (Slice 1) scored the lowest energy consumed regardless of the number of devices. This returns to the reliability utility value that forces delay-sensitive devices to take the most reliable path with the lowest spreading factor values and transmission power compared to MCC (Slice 2) and LCC (Slice 3). On the other hand, considering load in utilities will push sometimes a device to select a farther gateway with lower reliability. In that case, the device is configured with a higher spreading factor and will increase afterwards its transmitting power to reach the gateway efficiently. Hence, energy consumption for this device and other MTCD's belonging to the same slice will also increase on the expense of reliability. This explains the highest energy consumption on LCC slice members because they do not consider reliability in their metric calculations.

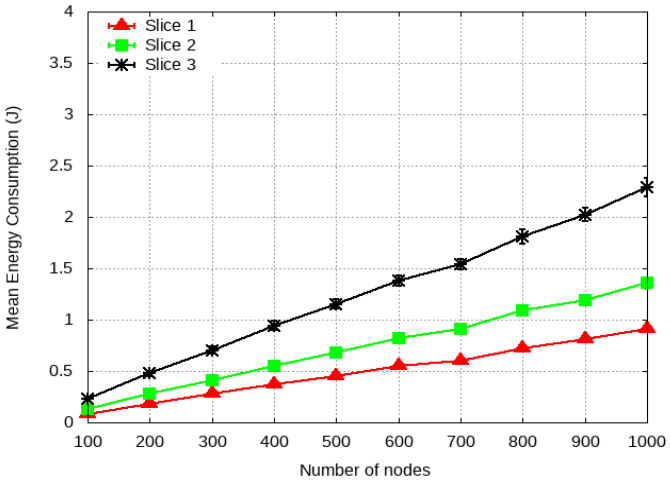


Fig. 4. Energy Consumption Variation

Simulation Parameters	
Simulation Time	600 seconds
Slicing Interval Time	50 seconds
Cell Radius	7.5 KM
Number of replications	50
MTCDs and GWs distribution	Random Uniform
Propagation loss model	Log-distance
Bandwidth	125 kHz
Spreading Factor	{7,8,9,10,11,12}
Confidence intervals	95%
European ISM sub-band	863-870 MHz
Power Consumption Parameters [14] [24]	
Battery Maximum Capacity	950 mAh
LoRa Supply Voltage	3.3V
Amplifier Power's added Efficiency	10%
Connected (Tx/Rx-SF7)	2 dBm
Connected (Tx/Rx-SF8)	5 dBm
Connected (Tx/Rx-SF9)	8 dBm
Connected (Tx/Rx-SF10)	10 dBm
Connected (Tx/Rx-SF11-12)	14 dBm
Standby	0.09 mW
Sleep	0 mW

TABLE III
SIMULATION PARAMETERS

B. Dynamic Slicing (DS) vs Fixed Slicing (FS)

To validate our dynamic-slicing proposition we consider the scenario where we fix the number of receive paths assigned in an equal manner between slices for all slicing intervals and we compare it to the proposed dynamic slicing where path reservation is dynamically done based on the MLE throughput estimation for each slice. We evaluate the performance of both mechanisms in terms of reliability, throughput and the percentage of devices that respected their delay deadlines. Simulations are replicated 50 times to be able to draw figures with 95% confidence intervals.

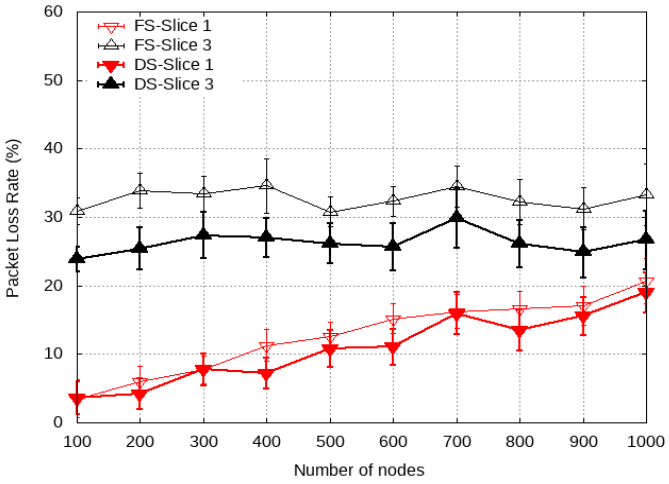


Fig. 5. Packet-loss Rate Variation

*1) **Packet Loss Rate (PLR) Variation**:* As plotted in **Fig. 5** below, excluding load from metric calculations of HCC slice reduced the percentage of packets lost due to reliability consideration unlike the case of LCC slice where only load was considered. Moreover, regarding the latter comparison, one could obviously note the efficiency of dynamic slicing in reducing the rate of packets lost with an approximately 50% improvement in both HCC and LCC slices. This improvement returns to the proper estimation method which adapt the bandwidth of a slice to the needed throughput starting by the most critical slice. The sporadic nature of packet transmissions in HCC slice require low latency and high reliability with unsteady throughput needs which may sometimes exceed the actual bandwidth reserved on the most reliable gateway. Hence, if this slice do not get the required bandwidth, PLR increases due to congestion.

*2) **Throughput Variation**:* In **Fig. 6**, increasing the number of devices decreases throughput to small values in the order of few kb/s. This is normal due to the increasing congestion in each slice. However, regardless of the congestion, DS method had the upper hand in all the slices simulated. One should note that as expected, HCC slice always scored the highest throughput compared to MCC and LCC slices due to the lowest spreading factor configured on its members when choosing the highest reliable paths between the ones available on each LoRa gateway.

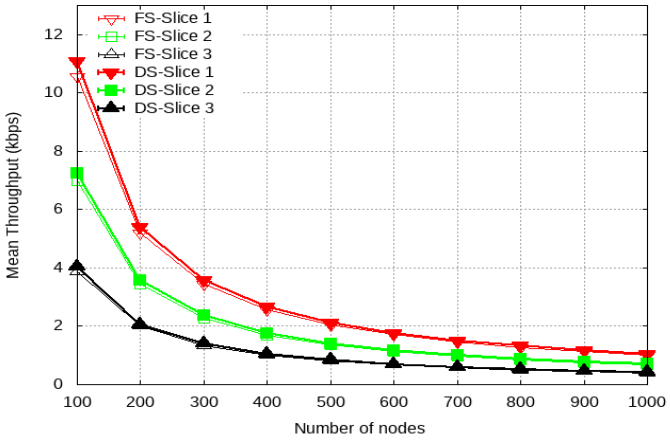
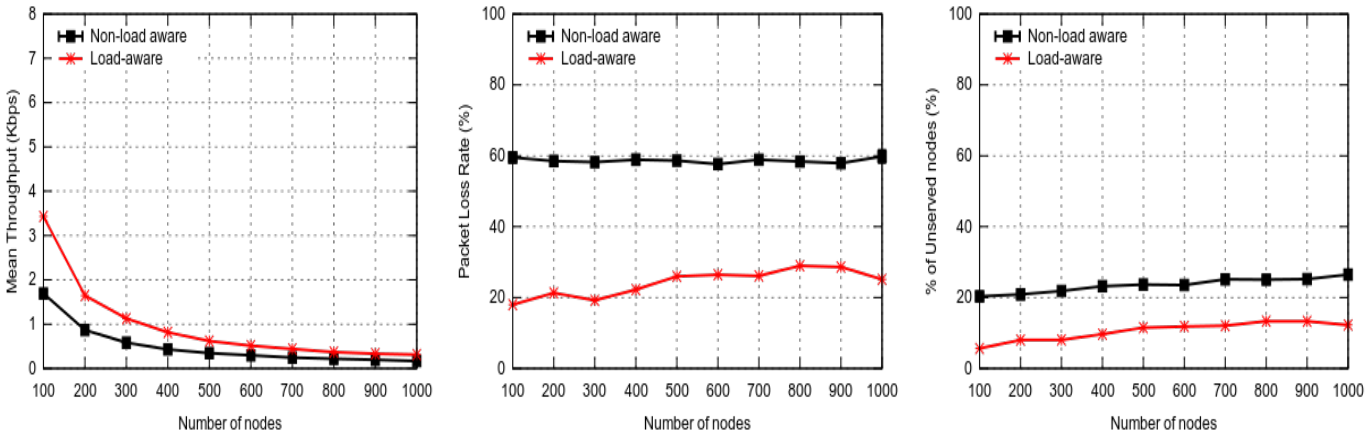


Fig. 6. Mean Throughput Variation

*3) **Percentage of Unserved nodes**:* The efficiency of DS is mainly shown in **Fig. 8** below. Here, LCC slice was not simulated due to the delay-tolerant nature of its members. An IoT device transmitting on a more reliable link is configured with a spreading factor that will let him reduce the spectrum occupancy time. This explains why HCC slice members always respected their delay deadlines compared to MCC members regardless of the strategy adopted for slicing. The improvement in PLR that DS achieved compared to FS strategy, reduced the number of packets retransmitted and the percentage of devices that respected their delay deadlines. Therefore, in a comparison between HCC and MCC slice and despite having the lowest packet delay budget, HCC slice members with DS always scored better results than FS in



(a) Mean Throughput Variation (b) Average Packet-loss Rate Variation (c) Percentage of Unserved Nodes Variation

Fig. 7. Performance Study with/without considering load in metric calculations

respecting delay deadlines with an unserved rate that never exceeded 20% of the total number of packets transmitted.

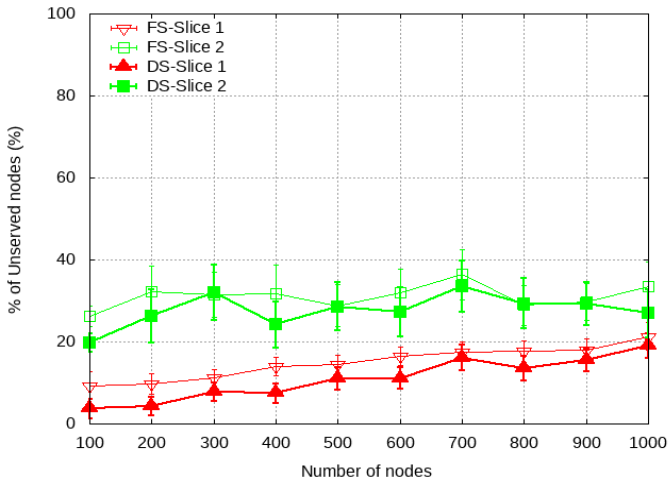


Fig. 8. Percentage of Unserved nodes

C. Load vs Non-load aware

We also studied the impact of adding load metric to utility calculations when we increase the number of LoRa devices assigned to each slice. In this scenario, we focused on evaluating performance metrics in terms of throughput, packet loss rate and the percentage of devices that respected their delay deadlines with (load-aware) and without (non-load aware) considering load in utilities computation. In an IoT environment, nearby devices may have interests with the same gateway regardless of the slice that they occupy. In that case, all devices will try to connect with the same gateway's bandwidth which will suffer from congestion specially if they belong to the same slice decreasing with it the probability of successfully decoding the packet. Therefore, it would be better if the algorithm tries to decrease the load by distributing and forwarding packets to a less-loaded slice on another gateway with acceptable reliability. Simulation results in **Fig. 7** prove the efficiency of load consideration when computing the mean values of slices with and without considering load in metric calculations. This mainly improved the performance of LoRa devices in terms of throughput **(Fig. 7a)** where a higher throughput was always achieved when the number of devices increase. This mainly returns to the 50 % improvement in the rate of packets lost shown in **(Fig. 7b)**. The centralized scheduler assign a device to a less-loaded gateway when the latter suffers from congestion and this mainly happens for MCC and LCC slices. HCC slice members get the highest priority and the needed bandwidth on the most reliable gateway leaving a smaller bandwidth for the remaining slices and a higher probability of congestion. The number of devices that respected their delay deadlines also improved as shown in **(Fig. 7c)**. In this aspect, MCC slice is the main beneficiary from load consideration because it has delay requirements with lower capacity reserved for its members compared to than the one provided for HCC slice. This will increase the rate of packets successfully delivered for MCC slice members with respect to delay deadlines when they transmit to less-loaded gateways instead of loosing them cause of congestion.

V. CONCLUSION

The number of technologies enabling LPWANs are increasing with the goal to improve QoS in an IoT environment with emerging technologies such as network slicing. In this paper, we evaluate the latter in LoRa technology with the goal of defining LoRa slices and maximizing utilities in each slice. Therefore, we propose a dynamic network inter-slicing algorithm based on a maximum likelihood estimation and an intra-slicing algorithm that meets the QoS requirements of each slice. Numerical results show that dynamic slicing improved the efficiency of LoRa devices in terms of throughput, energy consumption and the percentage of satisfied devices with regard to their delay requirements.

REFERENCES

[1] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5g: Survey and challenges," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 94–100, 2017.

[2] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, "End-to-end network slicing for 5g mobile networks," *Journal of Information Processing*, vol. 25, pp. 153–163, 2017.

[3] A. E. Kalør, R. Guillaume, J. J. Nielsen, A. Mueller, and P. Popovski, "Network slicing for ultra-reliable low latency communication in industry 4.0 scenarios," *arXiv preprint arXiv:1708.09132*, 2017.

[4] Y. Gadallah, M. H. Ahmed, and E. Elalamy, "Dynamic lte resource reservation for critical m2m deployments," *Pervasive and Mobile Computing*, vol. 40, pp. 541–555, 2017.

[5] P. H. Rezende and E. R. Madeira, "An adaptive network slicing for lte radio access networks," in *Wireless Days (WD), 2018*. IEEE, 2018, pp. 68–73.

[6] Y. Hao, D. Tian, G. Fortino, J. Zhang, and I. Humar, "Network slicing technology in a 5g wearable network," *IEEE Communications Standards Magazine*, vol. 2, no. 1, pp. 66–71, 2018.

[7] C. Delgado, M. Canales, J. Ortín, J. R. Gállego, A. Redondi, S. Bousnina, and M. Cesana, "Joint application admission control and network slicing in virtual sensor networks," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 28–43, 2018.

[8] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, and T. Watteyne, "Understanding the limits of lorawan," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 34–40, 2017.

[9] M. Cattani, C. A. Boano, and K. Römer, "An experimental evaluation of the reliability of lora long-range low-power wireless communication," *Journal of Sensor and Actuator Networks*, vol. 6, no. 2, p. 7, 2017.

[10] D. Magrin, M. Centenaro, and L. Vangelista, "Performance evaluation of lora networks in a smart city scenario," in *Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

[11] J. C. Zuniga and B. Ponsard, "Sigfox system description," *LPWAN@ IETF97, Nov. 14th*, 2016.

[12] R. Ratasuk, B. Vejlgaard, N. Mangalvedhe, and A. Ghosh, "Nb-iot system for m2m communication," in *Wireless Communications and Networking Conference (WCNC), 2016 IEEE*. IEEE, 2016, pp. 1–5.

[13] I. . W. Group *et al.*, "Ieee standard for local and metropolitan area networkspart 15.4: Low-rate wireless personal area networks (lr-wpans)," *IEEE Std*, vol. 802, pp. 4–2011, 2011.

[14] N. Blenn and F. Kuipers, "Lorawan in the wild: Measurements from the things network," *arXiv preprint arXiv:1706.03086*, 2017.

[15] C. Goursaud and J.-M. Gorce, "Dedicated networks for iot: Phy/mac state of the art and challenges," *EAI endorsed transactions on Internet of Things*, 2015.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[17] ——, "Birch: A new data clustering algorithm and its applications," *Data Mining and Knowledge Discovery*, vol. 1, no. 2, pp. 141–182, 1997.

[18] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, "A survey of clustering algorithms for big data: Taxonomy and empirical analysis," *IEEE transactions on emerging topics in computing*, vol. 2, no. 3, pp. 267–279, 2014.

[19] O. S. Vaidya and S. Kumar, "Analytic hierarchy process: An overview of applications," *European Journal of operational research*, vol. 169, no. 1, pp. 1–29, 2006.

[20] S. A. AlQahtani, "Analysis and modelling of power consumption-aware priority-based scheduling for m2m data aggregation over long-term-evolution networks," *IET Communications*, vol. 11, no. 2, pp. 177–184, 2017.

[21] B. Rogier. (2016) Network performance : Links between latency throughput and packet loss. [Online]. Available: https://www.performancevision.com/blog/network-performance-links-between-latency-throughput-and-packet-loss/

[22] (2018, March) Ns-3.28 documentation.

[23] A. Augustin, J. Yi, T. Clausen, and W. M. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.

[24] C. Bouguera, Diouris and Andrieux, "Energy consumption modeling for communicating sensors using lora technology," in *IEEE Conference on Antenna Measurements and Applications (CAMA)*. IEEE, 2018.