Received 29 August 2024, accepted 11 September 2024, date of publication 23 September 2024, date of current version 7 October 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3465634

RESEARCH ARTICLE

Accelerating Network Resource Allocation in LoRaWAN via Distributed Big Data Computing

PIETRO SPADACCINO[1,2], DOMENICO GARLISI[2,3], (Member, IEEE), ANDREA FRANCESCHI[1], ILENIA TINNIRELLO[2,4], (Member, IEEE), AND FRANCESCA CUOMO[1,2], (Senior Member, IEEE)

[1]Department of Information Engineering, Electronics and Telecommunications (DIET), Sapienza University of Rome, 00184 Rome, Italy
[2]Consorzio nazionale interuniversitario per le telecomunicazioni (CNIT), 43124 Parma, Italy
[3]Department of Mathematics and Informatics, University of Palermo, 90123 Palermo, Italy
[4]Department of Engineering, University of Palermo, 90128 Palermo, Italy

Corresponding author: Pietro Spadaccino (pietro.spadaccino@uniroma1.it)

This work was supported in part by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" through the Watering IoTs (WITS) and Supporting Project to Restart on Intelligent Networks for Telecommunications (SPRINT) focused project under Grant PE00000001—Program "RESTART"—CUP E83C22004640001.

ABSTRACT LoRaWAN is a Low Power infrastructure for the Internet of Things (IoT) with a centralized architecture where a single node, the network server, handles all data collection and network management decisions. Given the proliferation and widespread adoption of IoT devices, it becomes essential to incorporate Big Data paradigms at the network server to efficiently manage the enormous volumes of data. In this paper, we introduce a distributed and high-performance methodology for resource allocation in dense LoRaWAN networks, addressing the scalability issues that arise when processing large amounts of information from IoT devices, such as radio link quality. Our contributions establish the groundwork for a distributed implementation of the EXPLORA-C allocation strategy, capable of efficiently operating in large-scale networks. We present two approaches for implementing this distributed scheme: the Multi-Thread (MT) scheme and the Fully-Distributed (FD) scheme. Furthermore, we demonstrate the feasibility of this distributed implementation on top of the NebulaStream stream-based end-to-end data management platform. To validate the proposed approach, we exploit our co-simulation framework, EXPLoSIM, where the distributed implementation is fed with data from a simulated LoRaWAN network. This validation shows significant savings in execution time, latency, and scalability. Additionally, we generalize the concept by decomposing a centralized data aggregation scheme into a chain of stream-processing operators, which can be dynamically allocated across device, Edge, and Cloud levels. In the best scenario, our approach improves metrics such as execution time and data reduction by over 90% when compared to its centralized operation.

INDEX TERMS Big data, edge computing, fog computing, IoT, LoRa, LoRaWAN, LPWAN, stream data.

I. INTRODUCTION

The Internet of Things (IoT) is the new paradigm envisioned as a global network of devices capable of gathering information from systems and environments and interacting with each other. In this field, Low-Power Wide-Area Network (LPWAN) technologies have emerged as a viable alternative to traditional cellular technologies to provide power-efficient and cost-effective wide area connectivity for the IoT, especially suitable for smart metering applications. Indeed several countries, like Europe, target at connecting hundred of millions of smart meters for electricity as well as for gas and water utilities. This will have a direct impact on smart metering and will improve efficiency, with obligation schemes set for utility companies to achieve energy/gas/water savings.

The associate editor coordinating the review of this manuscript and approving it for publication was Huan Zhou.

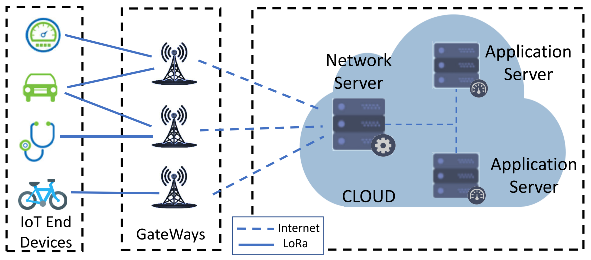


FIGURE 1. LoRaWAN network architecture.

The ETSI technical report on LPWAN Chirp Spread Spectrum [1], containing market information related to scalability and technical details of the system, foresees that, in few years, the IoT devices density will reach 5582 devices per square kilometer and 3.5 gateway per square kilometer in an urban scenario. Beside, the IoT Analytics forecast [2] indicates LPWAN as the fastest growing IoT connectivity technology over the next years, with two dominant technologies, namely LoRaWAN and Sigfox [3], accounting for about 86% of the total devices.

This study focuses on LoRaWAN [4], a complete network solution built on top of LoRa, a physical layer technology patented by Semtech [5]. While there exist already several papers dealing with Big Data approaches to process, at the application level, the data gathered by IoT devices, as well described in the survey in [6], few papers dealt with the application of Big Data solutions to the management of the IoT network, for example for dealing with the problem of resource allocation. Indeed, with IoT, not only the application data become big, but also the network management ones.

A. LoRaWAN ARCHITECTURE

LoRaWAN defines a complete network solution: this specification is publicly available and it is promoted by the open-source LoRa Alliance [4]. Figure 1 presents the LoRaWAN network architecture. Packets sent by End Devices (EDs) are collected by GateWays (GWs) that are deployed in the covered area. Packets are forwarded from the GWs to the Network Server (NS), which is responsible to process them, forward related information to the IoT applications and store the collected data. The related high level information is sent to the Application Server (AS). In this architecture the network management is performed at the NS level.

GWs have the duty to forward not just the application data, but also a series of control information related to the management of the network. Such information may include the timestamp, the Received Signal Strength Indicator (*RSSI*), Signal to Noise Ratio (SNR) and other physical layer information. They are needed for functions such as dynamic data rates allocation, downlink operation, etc. The official implementation of the gateway logic provided by Semtech defines 12 different metadata entries for each forwarded application packet. The raw size in bytes of all such metadata combined can very well be larger than the single application packet in question, especially if the optimization strategy considers the radio channel traffic generated also from EDs belonging to different networks.

B. MOTIVATIONS FOR THE PROPOSED APPROACH

Our work focuses on a critical aspect of LoRaWAN network management, which is the dynamic selection of data rates for end devices (EDs). In this context, the LoRaWAN standard introduces the Adaptive Data Rate (ADR) algorithm as a fundamental mechanism. However, our main objective is not to introduce a new algorithm but rather to propose innovative solutions for effectively managing algorithms in the ADR family. Hence the novelty lies in our management approach, not in the algorithm itself. Generally speaking the algorithms in the ADR family define how the NS controls the wireless access parameters like data rate, transmission power, channels, etc. These algorithms collects link level information about quality parameters relevant to EDs. The central NS uses the data to select the most appropriate settings for each device.

Among multiple algorithms available in the literature that can be classified as belonging to the ADR family, we pick up EXPLoRa-C in [7], which has been optimized by taking into account the presence of multiple GWs. In particular, EXPLoRa-C takes into consideration the channel capture effect deriving by the LoRa modulation, mostly in multi-GW setting, and load balancing considerations among different data rates. The capture effect is represented when the probability of successful decoding one of two overlapped packets depends on the relative power difference between them. In LoRa this accrues in case of lower power difference in comparison with other modulation technologies. In the paper [7] it has been extensively shown the improved effectiveness of EXPLoRa-C with respect to the currently employed ADR algorithm, in a wide set of scenarios.

To attain this performance, EXPLoRa-C needs a complete knowledge of the network, including also EDs belonging to different operators. Indeed, LoRaWAN presents a centralized architecture, where the central NS collects all the network monitoring information, runs the SF allocation algorithm and provides the device settings parameters. In this algorithm, when the number of EDs in the network grows, the processing can require high computational resources and a very long execution time, that can mine the network scalability. Indeed, operation mode for Class A EDs [4] provides short and close receive windows (typically 2 windows of 1 second) after each ED transmission (typically a few times per day) so processing execution time should be kept low to compute and enforce command in these short windows.

Our contribution is not focused on the evaluation of the ADR performance, but rather on the re-design of the data rate allocation scheme under a Big Data paradigm. Our aim is to move the processing and the optimization logic from the NS (Cloud) to GWs, or in general to an Edge level located close to GWs. We assume then to have some process capabilities at the GWs, when the hardware capabilities are sufficient to accommodate algorithms execution, or at the Edge node in other cases. Our vision is confirmed by the current literature relevant to the streaming process in Big Data for IoT applications [8]. The new paradigm goal is then to scale to thousands of queries and millions of sensors from which data are generated. The accessibility of sources under different constraints, as well as efficient access paths, requires solutions completely different from what today's big data processing systems provide. Query distribution on Edge layer will bring multiple performance optimizations, including a reduction of traffic volume on the internet backhaul. For instance, data from EDs belonging to different operators is not needed to be forwarded to the NS, but can be processed at the Edge.

The contributions of this paper are:

1) laying the foundations for a distributed implementation of the EXPLORA-C allocation strategy able to efficiently work in large scale networks;
2) providing two approaches for implementing this distributed scheme, namely the Multi-Thread (MT) scheme and the Fully-Distributed (FD) scheme;
3) demonstrating the feasibility of the distributed implementation of the chosen ADR scheme on top of a stream-based end-to-end data management platform called NebulaStream;
4) validating the savings in terms of execution time, latency and scalability of the distributed implementation of the proposed approach by exploiting the our co-simulation framework called EXPLoSIM, according to which the proposed implementation of the scheme is fed with data provided by a simulated LoRaWAN network.
5) generalizing the possibility of decomposing a centralized data aggregation scheme in a chain of stream-processing operators, which can be dynamically allocated at different device, Edge and Cloud levels.

The remainder of the paper is organized as follows. Section II presents the main related works. The proposed approach is presented in Section III. The implementation of the proposed approaches by exploiting NebulaStream is discussed in Section IV. Section V presents the architecture of the co-simulator, called EXPLoSIM, while the resulting behavior and performance analysis are discussed in Section V-C. Finally, Section VI concludes the paper.

II. RELATED WORKS

LoRaWAN works on scientific and medical (ISM) radio bands and following the frequency plan provided in [9]. For the European region, the applied frequency plan is EU863-870: it provides 8 channels, two available bandwidths (125 kHz and 250 kHz) and 6 symbol times also called Spreading Factors (SFs), leading to different data rates. The communication technology is based on a chirp spread spectrum modulation. Symbols are built by cyclically shifting chirp pulses, which span the entire frequency band within the symbol time. The SF defines two fundamental values: i) the duration of the symbol, which is given by an entire multiple N of a chip time T_c, with $N = 2^{SF}$; ii) the number of raw bits that can be encoded by that symbol, which is equal to SF. Finally, since the chip time T_c depends on the bandwidth, the combination of SF and bandwidth identifies the selected data rate. Finally, the SF in LoRaWAN directly impacts the coverage distance by improving signal processing gain and link budget. Higher spreading factors lead to increased coverage distances but come with trade-offs like reduced data rates and increased transmission times. Choosing the appropriate spreading factor depends on the network management, such as desired communication range, data rate, and energy constraints.

A. LoRaWAN NETWORK OPTIMIZATION PROBLEM

The problem of SF allocations in LoRaWAN networks is a problem of current interest, because SF allocations directly impact the network capacity and the energy consumption in large scale networks. Indeed, the higher the SF used by a given node, the more robust is the transmission in presence of interference, but the longer is the transmission time (also called time-on-air (*ToA*) and the relevant energy consumption. To maximize both the battery life of the EDs and overall network capacity, LoRaWAN can configure the SF (and the consequent data rate) of each ED individually by means of a data rate allocation scheme [10], [11], [12]. Semtech provides recommendations for implementing the ADR algorithm, which is adopted by different operators as well as "The Things Network", an open LoRaWAN network [13].

Since SF allocation has a dramatic impact on system-level performance, recent literature analyzed various algorithms for selecting SFs, while optimizing different performance figures such as the packet delivery probability and the energy consumption of the network devices [14], [15]. In [7], we provided a novel strategy, named EXPLORA-C for implementing a suitable SF allocation algorithm in LoRaWAN systems. The key idea is to assign the SFs to the EDs in a way that assures an equal channel occupancy time to all the available SFs, by taking into account the link budget constraints of each EDs. These features enable a LoRaWAN network to have a very high capacity and improve the network scalability. However, the approach requires a complete awareness of the radio channels, including the traffic generates by EDs belonging to a different network. Consequently, the total amount of generated data overloads the decisions taken by a central network server.

B. BIG DATA AND STREAM PROCESSING FRAMEWORK

In case of large IoT LoRaWAN systems, a high volume of data will reach the NS. Part of these data is directed to the ASs, but another big part may be used by the NS to accomplish network management and optimization. In this perspective we consider a LoRAWAN network as a Big Data system. The convergence between Big Data approaches and IoT solution has been already introduced in literature to promote the scalability of both the systems.

Traditional data management systems do not work properly with Big Data applications as they embrace either the Cloud or the Fog computing paradigm. Systems based on the Cloud paradigm do not exploit the full capabilities of IoT devices. To implement IoT applications, these systems require the collection of sensor data in a data center before processing. This centralized paradigm presents a bottleneck for upcoming IoT applications, which need to process data from millions of distributed sensors. On the other side, solutions based on the Fog paradigm may experience lack of storage, energy or computational resources when dealing with Big Data applications.

Among the many challenges of adopting edge and cloud computing is the issue of whether and how to separate or integrate data storage and services. In our paper, the service presented in this paper present a separation between data storage and services. The concept of Personal Data Stores (PDS) presented in [16] exemplifies this technological effort, leveraging distributed nature to enhance users' control over their data, enabling users to decide where and how their data are stored when accessing the required services.

Efficient data management in edge computing environments is increasingly critical, particularly with the rise of Internet-of-Vehicles (IoVs) applications that demand low-latency data access and reduced network congestion. Advanced edge caching strategies, such as those developed using Distributed Multi-Agent Reinforcement Learning in the DMRE and DeepDMRE frameworks [17] , have been shown to significantly enhance content delivery efficiency and minimize access costs by optimizing cache replacement and reducing redundant data transmission

In recent advancements within distributed systems, particularly concerning the integration of big data and edge computing, the concept of Distributed Continuum Systems (DCSs) has emerged as a crucial framework. Pujol et al. [18] discuss the importance of incorporating causal models into DCSs to ensure these systems are not only technically efficient but also socially responsible. They highlight how the deployment of DCSs, which extend beyond traditional cloud infrastructures to edge and IoT devices, requires dynamic adaptation while maintaining stringent performance and quality-of-service standards.

The Follow-Me AI concept in [19], which aims to improve interactions within smart environments, optimize energy utilization, and enhance data control, can also be applied to edge computing systems, such as those implemented in this paper. By deploying AI agents that accompany users, the system can manage data based on user consent and dynamically adjust environmental controls and computing resources to align with user preferences in edge computing scenarios.

Emerging distributed approaches try to balance processing operations performed at the Edge and at the Cloud, by applying innovative paradigms for reducing the data at the Edge, while leaving the most expensive computational operations at the Cloud. The idea is implementing some mapping functions at the network Edge for partially aggregating local data and returning only the mapping results to upstream reducers [20], with benefits also on security [21].

A large set of software frameworks have been designed or extended for dealing with this paradigm, among which Hadoop 2.0 [22], Spark [23] and Flink [24]. These frameworks are similar in terms of functionalities for optimizing the query process. The MapReduce programming model, as implemented for example by the Hadoop framework, is prevalent for many computing problems, including those that are not necessarily data-bound [24]. The Spark system utilizes Resilient Distributed Datasets [23], while Flink uses the DataStream and DataSet APIs to express streams and batching of data, respectively. Novel data processing platforms, like NebulaStream [8], address the heterogeneity, unreliability, and scalability challenges and enables effective data management for the IoT, aiming to unify the programming environments of Big Data and IoT. In particular, NebulaStream addresses unreliability by applying dynamic decisions and incremental optimizations during the runtime operation of the network.

In this work, we refer to NebulaStream, because it best fits the architecture of the IoT LoRaWAN scenario, which we report later in Section IV. In this architecture, the IoT data, including radio tap information, are routed from the sensors to the Cloud across the Edge layer.

III. BIG DATA APPROACHES FOR LoRaWAN ADR

The SF allocation in LoRaWAN networks has a key role, not only for guaranteeing that each ED can work with a suitable link-level budget, but also for improving the overall system capacity. Indeed, given the (semi) orthogonal nature of different SFs, the SF allocation can be exploited for balancing the load across the available SFs, thus maximizing the packet delivery probability.

We note that the ADR module within the Network Server operates transparently to the application layer. Modifications to this module do not impact the Network Server itself, nor do they affect the operation of the application. Our approach does not alter the Spreading Factor settings of the devices directly; rather, it refines the method by which the SF is calculated within the NS. The protocol for setting the SF on the devices remains unchanged, ensuring that our approach does not affect the LoRaWAN standard or its compliance. This means that the integration of our method with existing applications and infrastructures is seamless, with no need for adjustments to the current LoRaWAN implementations.

In this section, we present the problem of resource allocation in LoRaWAN networks as a big data problem. We first recall the centralized EXPLoRa-C algorithm, and then we

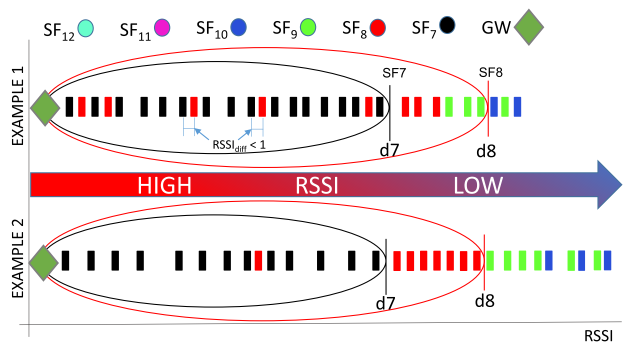


FIGURE 2. Two exemplary SF allocations performed by EXPLoRa-C with perfect (top row) and imperfect (bottom row) load balancing: each ED is represented by a different rectangle colored as a function of the assigned SF.

introduce two enhanced versions, named Multi-Thread (MT) and Fully-Distributed (FD) schemes.

A. EXPLORA-C CENTRALIZED OPERATION

To summarize the main behaviour of EXPLORA-C, let us consider the simplified scenario depicted in Figure 2 with 30 EDs (indicated by rectangles) placed within a cell. The link-budget constraints of the different modulation formats (or SF) available in LoRa are mapped into a maximum operating distance from the GW (indicated by a diamond shape). EDs placed in the inner area (whose distance d is lower than d_7) can work at any SF including SF_7, i.e., at the maximum data rate. Conversely, EDs in the region $d > d_7$ cannot employ SF_7; the constraint is more severe in the region $d > d_8$, where also SF_8 cannot be used.

The EXPLORA-C strategy aims at balancing the cell load among the SFs, while considering the link-budget constraints of the different modulation formats and the $RSSI$ (in dBm), measured by the EDs. Note that the load balancing criterion is implemented by also taking into account that different modulation formats lead to different packet transmission times. Even when nodes generate packets of the same size and with uniform rate, the number of devices that can be allocated at each SF is not the same, because the normalized offered load is equalized for a number of devices inversely proportional to the packet transmission time. For example, in Figure 2, taking as a reference the transmission time achieved at SF7, and assuming that each SF increment perfectly doubles the transmission time, the ideal allocation for achieving load balancing leads to 16 EDs at SF7, 8 EDs on SF8, 4 EDs at SF9 and only 2 EDs at SF10.

The exact proportion of nodes leading to perfect load balancing when all the nodes employ uniform source rates s_{sf} is presented in Table 1 together with the exact transmission times at different SFs.

EXPLORA-C has been implemented as a "sequential waterfilling" algorithm. As shown in the figure, for facilitating the selection of a data rate compatible with the link budget, EDs are ordered according to their maximum $RSSI_X$ value from the closest GW (from the highest to the lowest value). SF allocations are performed sequentially by verifying:

1) the link margin from the candidate SF;
2) the constraint on the maximum number of EDs on each SF;
3) the satisfaction of a diversity condition with the ED immediately preceding the current one in the ordered list.

TABLE 1. *ToA* (in *ms*) as a function of SFs when payload size is 20 byte and coding rate is 4/5; resulting percentages P_{sf} in accordance to optimal load allocation, orthogonal and not-orthogonal cases.

SF	7	8	9	10	11	12
ToA [msec]	49	91	165	330	659	1188
$\mathbf{P}_{sf}$ [%]	50.75	26.98	14.07	0.060	0.019	0.002

The goal of the last criterion is to increase the capture probability in case of collisions generated by EDs with similar $RSSI_X$ values: if the EDs are placed in different cell areas and therefore their neighboring GWs are not the same, they can be allocated on the same SF; otherwise, they are allocated on different SFs. Various diversity metrics between two EDs can be defined by considering the vectors of $RSSI$ values measured by all the GWs when receiving their packets, such as the euclidean distance between the vectors or the number of vector components whose difference is lower than a given threshold. In the single GW case shown in Figure 2, for example, EDs whose $RSSI$ values are too similar (i.e. whose position on the $RSSI$ scale is too close) are allocated on different SFs. In some cases, the constraints on the link-budget do not allow to implement a perfect load balancing, as shown in the bottom case of Figure 2. More details on EXPLORA-C can be found in [7].

EXPLORA-C is executed by the central NS in the Cloud, on the basis of the $RSSI$ values forwarded by the GWs when receiving packets from the EDs in their coverage area. The values are periodically averaged for associating a vector of $RSSI$ values to each ED. Decisions are taken in multiple steps, in which EDs are sequentially processed and allocated to a given SF or passed to a next decision step. Being M the total number of EDs and N the number of GWs, the scheme finds the maximum $RSSI$ value of each ED towards the GWs, sorts all EDs as a function of the best $RSSI_X$ value from the closest GW, and then sequentially processes EDs for finding a diversity metric towards the closest EDs in the sorted list. The resulting complexity is in the order of $O(M \cdot N) + O(M \cdot logM)$.

B. MULTI-THREAD SCHEME SOLUTION

As evident from the previous description, legacy EXPLORA-C works with a significant amount of data: in principle, all packets collected from an entire IoT network are processed and filtered at the NS. In order to reduce the complexity of data management operations, it is possible to split the data in independent segments, and allocating an algorithm thread to each data segment.

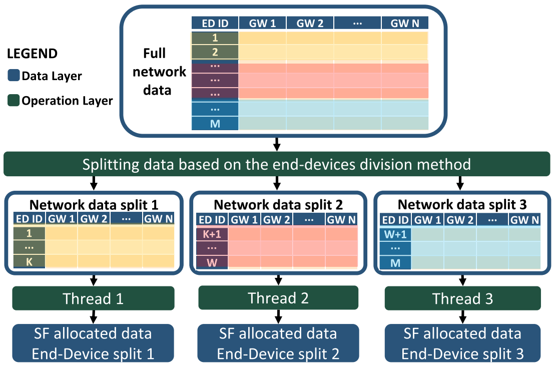

LEGEND
Data Layer
Operation Layer
Full network data
Splitting data based on the end-devices division method
Network data split 1
Network data split 2
Network data split 3
Thread 1
Thread 2
Thread 3
SF allocated data End-Device split 1
SF allocated data End-Device split 2
SF allocated data End-Device split 3

FIGURE 3. Multi-Thread flow diagram.

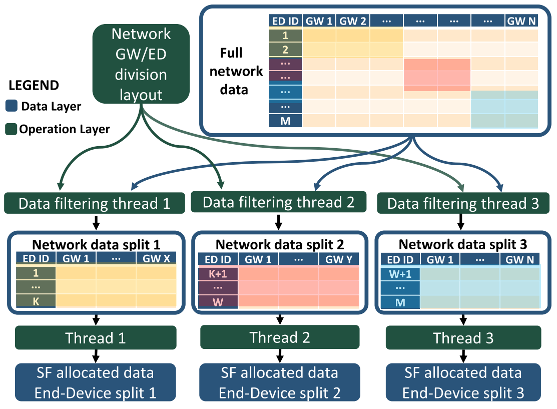


FIGURE 4. Fully-Distributed flow diagram.

Figure 3 depicts the data splitting and thread allocation, starting from the knowledge of the complete data set, which is represented in the top part of the figure as a list of EDs (numbered from 1 to M) and GWs (numbered from 1 to N). Indeed, we can envision a matrix data structure, in which for each couple (ED_i, GW_j) we store the average $RSSI$ (in a given time window) value measured at the $j-th$ GW from the reception of a packet sent by the $i-th$ ED. In case the $j-th$ GW is not able to receive packets from the $i-th$ ED, the value is set to $-\infty$. The splitting can be simply implemented by grouping the rows of the entire matrix $M \cdot N$ into G groups. In the case of Figure 3, in which $G = 3$, we then achieve three matrices, whose maximum dimensions are $\lceil M/G \rceil \cdot N$. A different thread is then activated for working on each sub-matrix. The output of the thread will be a set of SF allocations relevant for the group of EDs included in the data split. This solution can be also addressed as Multi-Thread (MT) scheme. Being $K = \lceil M/G \rceil$ the maximum number of EDs corresponding to each data split, the overall complexity of this scheme version is in the order of $O(K \cdot \log K) + O(K \cdot N)$. Since each thread has a complete view of the GWs seen by each ED, the computation of the diversity metric between devices is not affected by the data split. Therefore, although each thread will work with a smaller amount of EDs, provided that the group is big enough for including EDs experiencing heterogeneous channel conditions, the final SF allocation will lead to network performance similar to the ones achieved by the centralized version of the scheme.

C. FULLY-DISTRIBUTED SCHEME SOLUTION

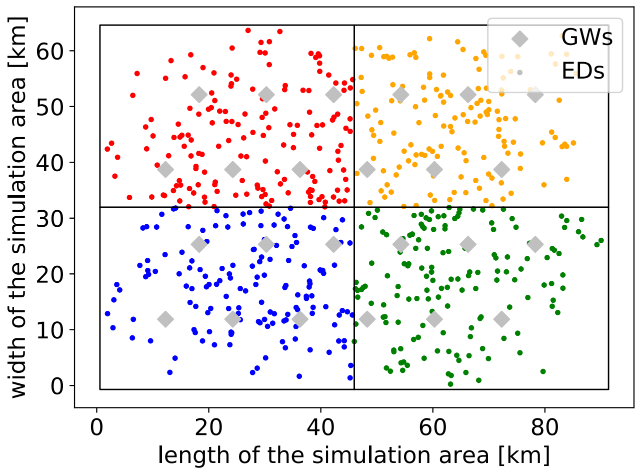
As a further step for reducing not only the complexity of the scheme, but also the data volume forwarded to the centralized NS, we can envision the possibility of allocating the execution of these threads at the network Edge. We assume that a set of GWs in physical proximity is managed by an Edge server that executes one thread. This solution can be also addressed as Fully-Distributed scheme, for indicating that threads refer to clusters of GWs. In such a case, as depicted in Figure 4, the complete matrix of data is not partitioned by splitting the rows, but it is reduced in smaller sets of rows and columns. Being K the maximum number of EDs assigned to each Edge server, the complexity is now $O(K \cdot logK) + O(K \cdot \lceil N/G \rceil)$, assuming that Edge servers work in parallel. However, in this case, the diversity metric between EDs will be based on a smaller set of GWs and therefore will be less accurate than in the previous case.

As highlighted in Figure 4 by the division layout module, different solutions can be envisioned for assigning a specific ED to a given Edge server (i.e., cluster of GWs). In case EDs are static and their geographical coordinates are known, one simple solution can be based on the definition of geographical bounds delimiting each cluster area. We call this method Geographical-Splitting. Alternatively, a belonging criterion can be based on the comparison between the best $RSSI_X$ value measured at a reference GW of the cluster and a minimum threshold. We name this method $RSSI$-Splitting. With this solution, it may happen that the same ED can be tackled by different clusters (in case the condition is verified for reference GWs belonging to different clusters) or by none of the clusters. Inter-cluster communications can be considered for solving this issue. By exchanging the list of EDs with neighbor clusters, it is possible to remove EDs whose best $RSSI$ value is measured in a different cluster, or add EDs for which the best $RSSI$ value is measured in the current cluster, even if lower than the admission threshold.

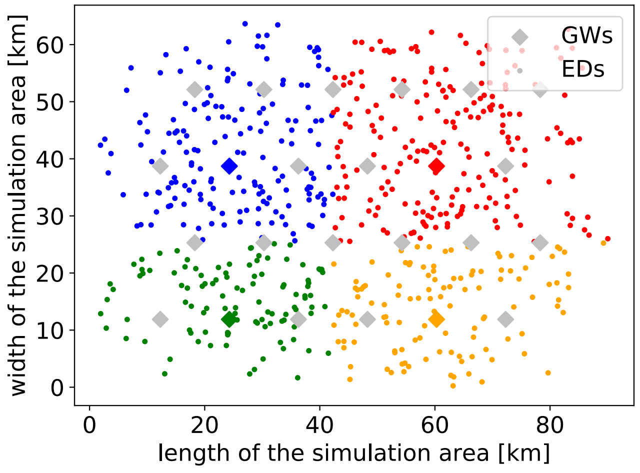
Figure 5 shows two examples of distributed clustering of EDs in a network with 24 GWs and 600 EDs. In Figure 5a, we consider four clusters delimited by the rectangles depicted in different colors on the network area. EDs are associated to the cluster by only considering the geographical coordinates. In Figure 5b, we consider four clusters whose reference GW is colored with a different color. EDs are assigned to the cluster with the best $RSSI$ value towards the reference GW, under the assumption that inter-cluster communications are implemented.

IV. IMPLEMENTATION BY DISTRIBUTED STREAM PROCESSING

The problem of end-to-end data management for IoT applications has been addressed by using different paradigms: the Cloud paradigm, in which all data are collected by a central data center before processing; the fog paradigm,



(a) Network scenario according to the Geographically-Splitting method.



(b) Network scenario according to the *RSSI*-Splitting method.

FIGURE 5. Network scenarios with different splitting methods.

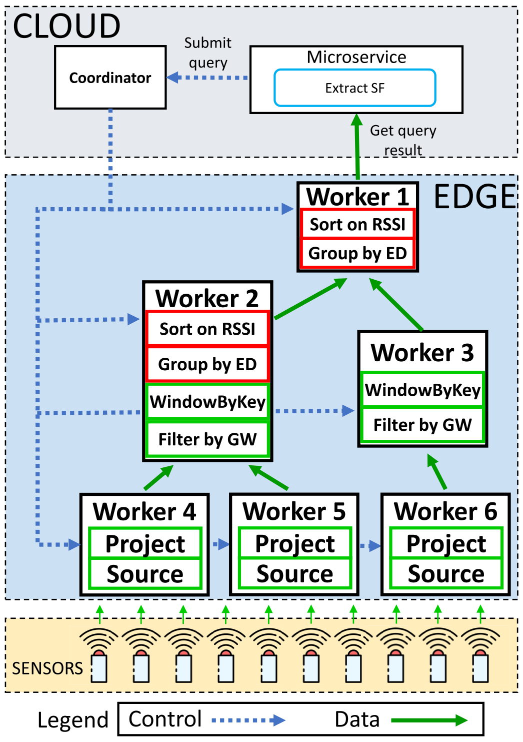


FIGURE 6. Example of network topology with distributed operators that implementing EXPLoRa-C Fully-Distributed algorithm.

in which Edge devices work on data aggregation for reducing the data volumes to be transmitted towards the Cloud; the emerging general purpose sensor-fog-cloud approach, in which data streams and processing operators can be dynamically allocated at different nodes along the path from the source to the Cloud. Since the problem of network management for LoRaWAN networks can be envisioned as a special IoT Big Data application, in this section we consider a possible implementation framework based on the latter sensor-fog-cloud approach. In particular, we consider the possibility of implementing EXPLoRa-C as a data-stream distributed processing on the *NebulaStream* platform [8].

A. END-TO-END PROCESSING PLATFORM

Figure 6 shows a general representation of the *NebulaStream* data processing framework, in terms of sensors, which produce data without processing them, and *workers* where data are processed along the path. *Workers* can be allocated at both the Edge and Cloud layer in a hierarchical structure. *Workers* can rely on heterogeneous processing and storage capabilities as a function of the node in which they are instantiated; moreover, *workers* close to the sensors process data locally, reducing the latency of the distributed application. The processing tasks and execution plans of the *workers* are orchestrated by a central *coordinator*. The *coordinator* is aware of the resources available at each node registered to the *NebulaStream* platform and takes decision on the workload distribution and query processing, by also considering the network topology (i.e. the data paths across the nodes). The figure shows both the control plane, for configuring the execution plans at the *workers* (dashed lines), and the data plane, for the routing of the data from the sources to the Cloud layer (bold lines). Differently from pure Cloud-based systems or fog-based systems, *NebulaStream* allows optimization, scaling and load balancing across heterogeneous nodes and in presence of unreliable nodes and dynamic network topologies.

Consider now the mapping between the *NebulaStream* architecture and a network management application for LoRaWAN networks. The EDs deployed in the network act as a sensor layer, which transmit data packets for heterogeneous services (e.g. smart metering data, occupancy status of a parking slot, position tracking of a bycicle, etc.). The data are received by the GWs, which in the *NebulaStream* topology are called entry nodes. The GWs add physical layer information, such as the RSSI value, to the data packets received by the EDs before forwarding them to the upper layer. In most cases, even very simple GWs can act as *workers* for connecting to the data streams and doing simple operations (selection of packet fields, filtering) before forwarding data. Intermediate routing nodes can be deployed for aggregating data streams from multiple GWs. These nodes act as Edge nodes, where other *workers* can be installed for performing distributed processing operations.

Note that each ED in the sensor layer is connected to at least one node in the Edge layer, which is responsible to get the stream. Nodes on the path from the EDs to the Cloud can process only data routed through them, and therefore can only perform operations that do not require a full view of the system. Conversely, the Cloud layer is able to apply processing operations which require a global view of the system.

B. SCHEME DEFINITION WITH *NebulaStream* OPERATORS

In Figure 6, we are considering an exemplary implementation of the scheme on the *NebulaStream* platform, with a total number of 6 workers. We also assume that GWs have limited hardware resources and that the storage and processing capabilities of Edge nodes increase significantly with the hierarchical level of the nodes (i.e. in each hop towards the Cloud layer). The figure also shows the operators that have been identified for data pre-processing and for supporting the MT variant of EXPLoRa-C. Generalizations to the FD variant are straightforwards.

In the Edge layer, since workers 4, 5 and 6 are executed on LoRaWAN GWs with limited capabilities, the coordinator has configured very simple tasks: the *source* operator to connect the data stream and the *projection* operator for selecting a sub-group of data packet fields.

The output data are then routed through the next network node, which receives data streams from different GWs and apply the following tasks: i) filter packets through the *filter* operator, ii) segment data streams into different sub-sets through the *WindowByKey* operator (applied on a temporal basis); iii) apply grouping operation to extract cumulative features in a sorted way through the *Group by* operator. After the segmentation phase, we start the distributed implementation of the proposed scheme. These operators are computationally heavier and thus can be executed by a smaller number of Edge nodes.

For example, in the figure we are assuming that worker 3 will forward the segmented data-streams to worker 1, while worker 2 is able to execute the initial algorithm steps. The steps involve a *group by* operator, which is responsible of grouping all data referring to the same ED provided by multiple GWs, followed by a *sort* operator, which is responsible of ordering the data as a function of the strongest *RSSI* value. Finally, at the Cloud layer, data related to all the EDs in the network are merged for creating a complete data structure, which is then passed to the Cloud responsible of executing the aforementioned threads.

Consider now an exemplary query executed by using the *NebulaStream* platform, whose logic is reported in the flowchart in Figure 7. The process begins with ingesting

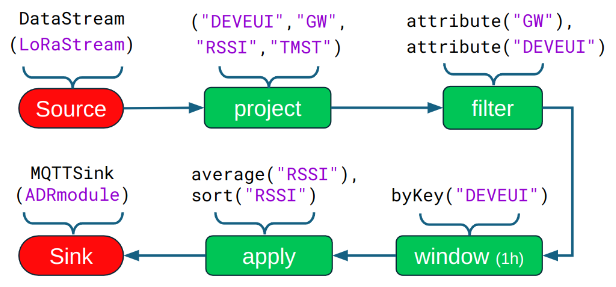


FIGURE 7. Flowchart of the query executed by *NebulaStream*.

the LoRaStream, which includes key attributes such as the device identifier (DEVEUI), gateway, RSSI and reception timestamp. The query then selects relevant data, filtering records selecting only the packets generated by the relevant devices and gateways, and organizes the data into one-hour time windows for temporal analysis. Within these windows, the system calculates radio metrics, such as RSSI, which are sent to the ADR module via an MQTT sink.

The benefit of the *NebulaStream* platform is mainly due to the distribution of the computational weight. Indeed, by distributing the grouping and sorting operators across the different Edge nodes, the computational weight of the algorithm is also distributed. As explained in Section III, the computational weight of the centralized version is $O(M \cdot N) + O(M \cdot logM)$ which then decreases in the MT and FD versions with the distribution of EDs and GWs. The $O(M \cdot N)$ factor of the weight equation is given by the search for each EDs of the GW associated with the maximum *RSSI* and it is reduced to $O(1)$ ahead of the distributed operators on the Edge nodes. This means that the remaining part of the algorithm executed in the Cloud layer has a computational weight of $O(M \cdot logM)$.

We remark that the topology in Figure 6 represents a fundamentally new end-to-end data processing platform. Indeed, data processing and operator placement are required to be network-aware: the coordinator monitors the network nodes and link capacities before applying an optimization strategy, in terms of data routing and task allocation, for reducing data volumes as early as possible, together with the application latency.

V. EXPERIMENTAL RESULTS

For validating the proposed approach, we developed a co-simulation tool, named EXPLoSIM, that integrates three different software components, as shown in Figure 8. The code used for the experiments is made public through Github.[1] The first component is the well known LoRaSIM simulator, an event-based simulator written in Python on the basis of the SimPy library [25]. The tool is used for creating different network scenarios in terms of node placements, traffic sources, and channel models, and for simulating the performance of the LoRa random access protocol under different SFs allocations. The second component includes

[1]https://github.com/netlab-sapienza/EXPLoRa-distributed

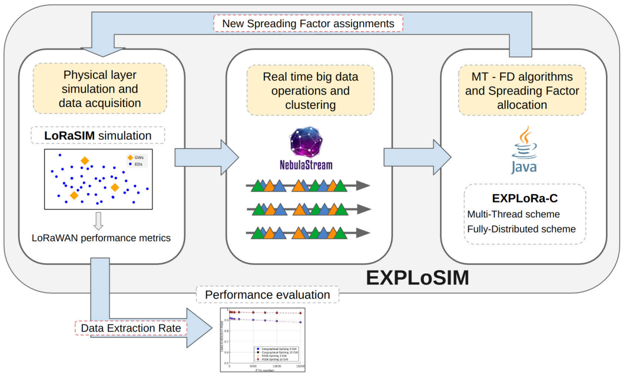

New Spreading Factor assignments
Physical layer simulation and data acquisition
LoRaSIM simulation
LoRaWAN performance metrics
Real time big data operations and clustering
NebulaStream
MT - FD algorithms and Spreading Factor allocation
Java
EXPLoRa-C
Multi-Thread scheme
Fully-Distributed scheme
EXPLoSIM
Performance evaluation
Data Extraction Rate

FIGURE 8. EXPLoSIM architecture.

TABLE 2. Simulation parameters.

Parameter	Value
Carrier Frequency - CF (MHz)	868.0
Bandwidth - BW (kHz)	125
Code Rate - CR	4/5
Packet size (byte)	20
Sending interval (mean)	4800 ms
EDs distribution	uniformly and not uniformly distributed
TXPower - TP	14 *dBm*
Capture Effect (coll.)	enabled
Distance GW-GW (m)	12000
Path loss at distance d, $L_{pl}(d)$	$L_{pl}(d_0) + 10\eta \log(\frac{d}{d_0}) + \chi_\sigma$ in dB, where χ_σ normal distribution with zero mean and σ^2 variance and $\eta = 2.9$, $\sigma^2 = 7$, $\overline{L_{pl}}(40m)$ $-66\ dB$

the NebulaStream framework, which performs big data operations. The final component is the Java software that implements the MT and the FD algorithms proposed in this work. The resulting SF allocation of the algorithms is given as an input to LoRaSIM to perform a new simulation of the LoRaWAN environment. We select Java since the resulting implementation is ready to interact with NebulaStream to submit the query and to receive the output result. The Java Threads work on data matrices of *RSSI* values which are artificially generated by the LoRaSIM simulator and collected by NebulaStream workers. In turns, decisions about SF allocations taken by the Java Threads are passed to LoRaSIM before simulating the LoRa random access protocol and computing the overall data extraction rate. In order to allow the LoRaSIM and the NebulaStream to communicate, we used MQTT message broker interface.

More into details, the LoRaSIM module is responsible of placing N LoRa nodes in a 2-dimensional space, where M LoRa GWs are deployed. Each LoRa node has a specific communication model, identified by the transmission parameters, such as the Transmission Power (TP), Carrier Frequency (CF), bandwidth (BW), coding rate (CR) and SF. The SF adopted by each node is decided by the output of the MT and FD algorithms.

LoRaSIM continues its execution for simulating the behavior of the whole network on the basis of the SF allocations provided by the algorithm and traffic model. The traffic model is defined by the sending interval and packet size, which is assumed to be constant for all the nodes. All the simulation parameters are summarized in Table 2.

A. EXPLoSIM CO-SIMULATION VALIDATION

For evaluating the effectiveness of the new EXPLoRa-C implementation, we consider the impact of the new versions of the scheme on the network-level Data Extraction Rate (DER). Indeed, this performance metric characterizes the capacity of LoRa systems, because it represents the number of messages correctly received by the backend system, i.e. by at least one GW in the network. We define the DER as the ratio between the number of packets successfully received by the NS and the total number of packets transmitted by EDs during the simulation time. Packets successfully received by multiple GWs are counted as a single non-duplicated packet. The DER depends on the number and position of EDs and GWs, as well as on the traffic source rate and SF decisions. DER is a value between 0 and 1; the closer the value is to 1 the more effective the LoRa deployment is. Since the MT and FD versions work with different data (and in particular, FD works with a limited vision of the network), the DER resulting from the two schemes could not be exactly the same.

We analyzed the performance of the MT implementations, considering two implementations working under both the Geographical-Splitting and *RSSI*-Splitting of EDs. Figure 9a reports the DER achieved by the MT version of the allocation scheme. The figure has been obtained for a number of EDs varying from 100 to 15000, which are deployed in an area in which the number of gateways is 3 or 10. From the curves, it is evident that the DER improves as the number of gateways increases. This expected result is due to the higher chances of experiencing a channel capture at some GWs, i.e. of correctly demodulating one of the colliding packets. From the figure, we observe that performance are not affected by the splitting criteria for EDs.

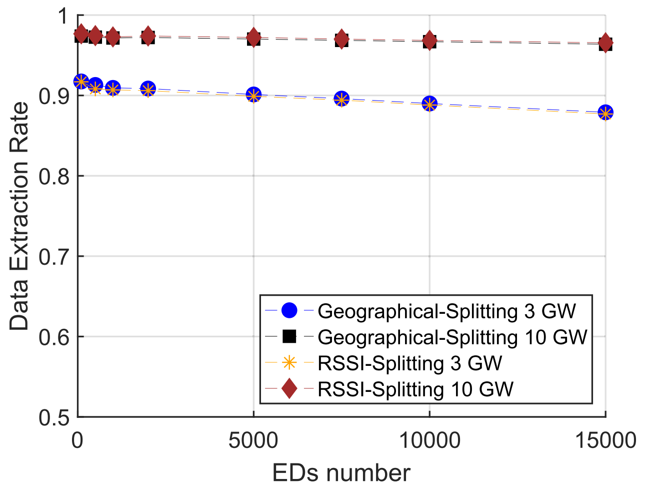
B. SYSTEM PERFORMANCE

In our implementation, we consider two different computing systems based on Intel x86 and ARM Arch64 technologies. The first system is typical of Network Servers and is referred as High Performance (HP) system, while the second one is typical of GWs or Edge Nodes and is referred as Low Performance (LP) system. Both the systems do not use GPU computing. More details about the configuration of memory, operating systems and execution environments are provided in Table 3.

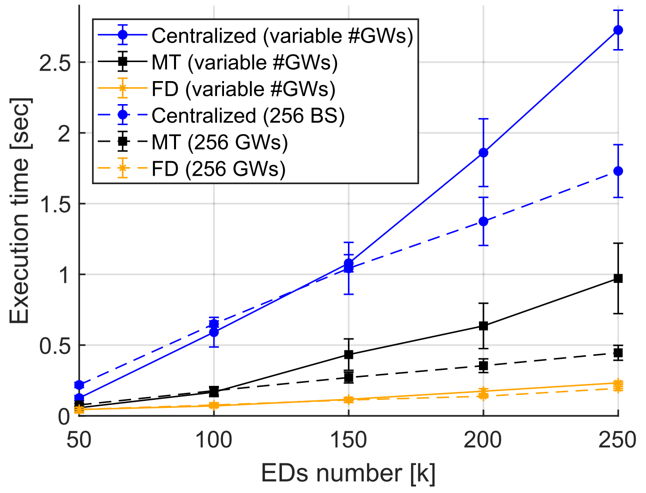
The each LP system is a Raspberry Pi device configured as GW-EDGE nodes. In total, our LP system comprise up to four independent Raspberry Pis. Each node is connected to a local network, simulating the behavior of actual LoRaWAN gateways with edge computing capabilities. These devices are responsible for handling the data streams from connected IoT devices, performing local processing, and participating in the distributed execution of the SF allocation algorithm.

TABLE 3. Configuration of the computing systems used for running EXPLoRa-C threads.

	LP Computing System	HP Computing System
CPU	Quad core Cortex-A72 (ARM v7l) SoC@1.5GHz	Intel(R) Core(TM) i7-7700HQ CPU@2.80GHz
Memory	4 GB	16 GB
OS	Linux RaspberryPi 5.10.60-v7l+	Linux
GCC	8.3.0	8.4.0
Python	3.7.3	3.6.9
Java	11.0.12 2021-07-20	16.0.2 2021-07-20
NebulaStream	0.5.103	0.5.103



(a) Comparison of the two different EDs splitting methods, Geographical-Splitting and *RSSI*-Splitting in terms of Data Extraction Rate.



(b) Performance comparison of the different approaches in terms of execution time in case of varying number of GWs (solid lines) and constant number of GWs (dashed lines).

FIGURE 9. Performance comparison in terms of algorithm execution time and data extraction rate.

To emulate a real-world scenario, the LP testbed was subjected to a continuous data stream generated by simulated IoT devices. This setup allowed us to monitor the performance of the key components involved in the distributed execution of the EXPLoRa-C algorithm. Specifically, we measured the execution time, bandwidth usage and final data reduction on edge and cloud devices, using a varying number of workers. We conducted the performance analysis by isolating and examining the individual macro components of our algorithm, particularly those that are the most computationally intensive within the Big Data framework. Analyzing these components separately rather than the entire application as a whole allows us to precisely identify and address the most resource-demanding operations of the distributed process. The results are reported in Table 4.

The results show that the our distributed algorithm is effective in managing the Big Data heavy computational tasks, with the Raspberry Pi devices successfully handling the distributed load while maintaining high performance levels. This demonstrates that even with the limited resources of Raspberry Pi devices, the algorithm can efficiently process the large volumes of data.

To address the scalability challenges observed in the initial testbed, we extended our evaluation using simulations. These simulations allowed us to model a larger LoRaWAN network with more gateways and a higher data throughput, conditions that would be difficult to replicate with the available hardware alone. In the next section we present quantitative results of the algorithm on the High Power system.

C. ALGORITHM SCALABILITY RESULTS

The proposed approaches aims at improving performance in terms of network scalability. For this reason, we consider not only the DER performance achieved, but also the improvements on the execution time under different assumptions about the computing power, the node density and the geographical distribution of devices.

According to the proposed approaches, the idea is to move the processing and optimization logic from the NS (Cloud) to an intermediate Edge layer (which is responsible of controlling multiple GWs in a cluster) or even to a single GW. Obviously, the allocation of the algorithm threads at different computing nodes implies different trade-offs between the processing capabilities available at the Cloud, Edge or GW level for the thread execution, and the volumes of input data to be transported towards different nodes. In particular, when the threads are executed by the GWs or by the Edge nodes the volume of backhaul traffic can be significantly reduced.

Table 5 compares the execution times achieved in the LP and HP computing systems for a network scenario

TABLE 4. Performance metrics of the Low Power (LP) system, with 187877 input records. The number of workers is the number of Raspberry Pis used in the system. Average IN and OUT are the average input and output traffic respectively.

Query	#output records	#Workers	Duration [s]	avg. IN [kB]	avg. OUT [kB]	Data Reduction on Edge	Final data reduction on Cloud
Group by node (10h)	1529	2	507.82	85.05	47.17	55.46%	99.19%
		3	326.57	69.12	37.06	53.61%	
		4	279.33	39.67	21.10	53.20%	
Group by GW (10h)	116	2	514.29	83.44	46.36	55.56%	99.994%
		3	315.71	47.25	25.35	50.07%	
		4	292.89	36.53	19.60	53.65%	
Filter	2670	2	449.69	90.70	1.37	98.49%	98.57%

TABLE 5. Execution time results for LP and HP computing systems as a function of the number of EDs and threads in milliseconds.

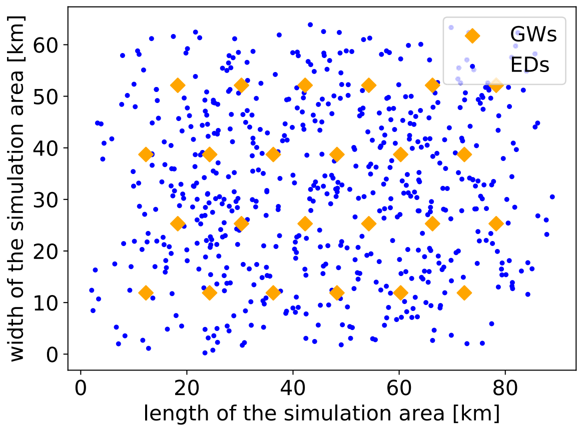
EDs number	1 000	10 000	100 000	1 000	10 000	100 000
Execution time on LP system [msec]	31.650	105.321	1026.242	24.213	91.323	740.072
Execution time on HP system [msec]	8.235	36.952	239.983	6.865	35.202	149.932
Number of threads	2	2	2	4	4	4

with 200 GWs and a varying number of EDs. In the table, we assume that the algorithm is executed with a number of threads equal to 2 (left most columns) or 4 (rightmost columns). As expected, the execution times depend on the scheme complexity and therefore get higher as the number of EDs increase, and can be reduced by working with multiple threads. On the LP computing system, we achieved an execution time longer than 1s for a network scenario with 100k EDs.

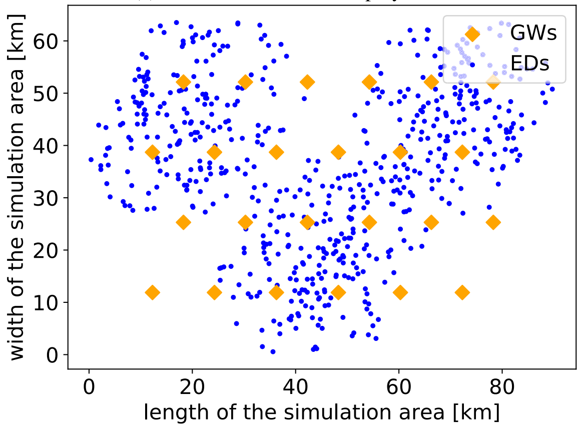
Figure 9b shows the execution times of different implementations (centralized, MT and FD with 8 threads) assuming to work with HP computing systems and different node density scenarios. Different colors and markers are employed for distinguishing the performance achieved under the three approaches. In particular, we assume that EDs vary from 50k to 250k (with steps of 50k), while the number of GWs is kept constant to 256 (dashed lines) or gradually increased from 100 to 500 with a step of 100 GWs (continuous lines). Indeed, to cope with an increasing traffic demand in LoRa networks, it is common to deploy more GWs. Error bars refer to the variability of execution times achieved in multiple execution runs and in particular to a confidence interval equal to 95% of the results. From the figure, we note that the FD approach significantly improves the execution time from 2.727s to 0.233s for the case with a variable number of GWs and from 1.730s to 0.194s for the case with a fixed number of GWs.

As shown in figure, the number of GWs has an impact on the algorithm complexity and consequent execution time, especially in the centralized and MT approaches. The different between the dashed and continuous curves is emphasized as the number of EDs increase. Indeed, when the number of EDs is 250k, the number of the GWs is 500 for the scenario with a variable number of GWs (solid line) and 256 for the scenario with fixed number of GW (dashed line).

We also analyzed the impact of the spatial distribution of EDs. In particular, we consider two types of deployments: a first uniform deployment (used for the previous evaluations), according to which EDs are randomly placed in the whole network area; a second clustered deployment, according to which EDs are concentrated only in some clusters of GWs. Figure 10 shows an exemplary comparison between the uniform (Figure 10a) and non-uniform

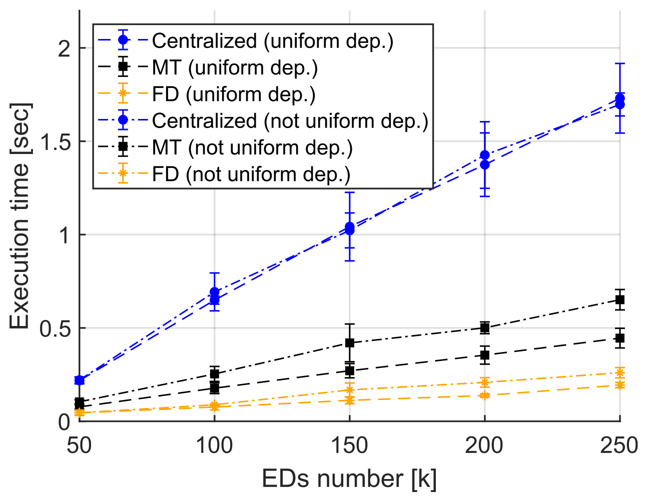


(a) Uniform ED location deployment.

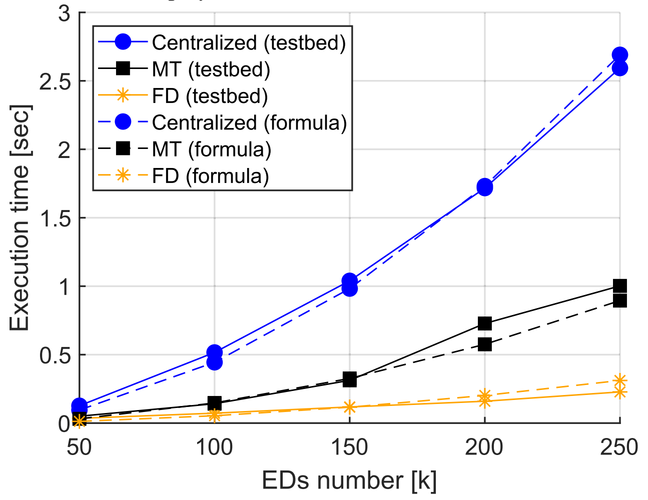


(b) Clusterized ED location deployment.

FIGURE 10. Example of different deployments for 600 EDs, uniform and not uniform.



(a) Execution time in case of EDs location uniformed deployed and not uniformed deployed.



(b) Execution time comparison between the theoretical algorithmic complexity and the testbed measurements.

FIGURE 11. Performance comparison of Centralized, MT and FD approaches in terms of execution time.

(Figure 10b) deployments, in a scenario with 24 GWs and 600EDs.

Figure 11a shows the performance comparison in terms of execution times, in a network with a fixed number of 256 GWs and uniform and non-uniform deployment of EDs. Different approaches are identified by different colors and markers, while error bars still refer to 95% of confidence intervals of results. Obviously, for the centralized case the execution times are not affected by the deployment of EDs, because all data are processed by the NS. For the other cases, working with non-uniform deployments of nodes degrade the execution times, because the computational load of each thread is no more balanced. In these conditions, performance depend on the most loaded thread, which requires an execution time longer than in the balanced case.

Finally, we perform a comparison between the theoretical execution time of the presented approaches and the complexity computation presented in Section III for a number of threads equal to 8 ($G = 8$). Figure 11b compares the measured execution times with the scheme complexity. For mapping the complexity into execution times, we empirically found a scale factor equal to 21 for the HP computing systems. From the figure, we note that our complexity evaluation perfectly corresponds to the execution times achieved when executing the different implementations of the proposed schemes, as discussed in Section III.

VI. CONCLUSION

In this paper, we have addressed the complexity of network management operations in LoRaWAN networks, specifically focusing on the challenges of spreading factor (SF) and data rate allocation for all end devices (EDs) deployed in the network. Our analysis highlights that managing such operations necessitates handling vast amounts of data, thus necessitating the application of Big Data approaches.

We examined the scalability challenges of the EXPLoRa-C resource allocation scheme, which employs a ‘‘sequential waterfilling’’ strategy requiring multiple steps to sort data records for all EDs. To enhance the scalability of this scheme, we proposed two extensions: the Multi-Thread (MT) and Fully-Distributed (FD) schemes. These extensions are designed to operate on both centralized and fully distributed processing platforms.

Our contributions also include a detailed analysis of potential performance degradation in multi-threaded executions. We demonstrated that multi-threading does not compromise the effectiveness of SF allocation in a centralized network server and has minimal impact in fully decentralized implementations. In decentralized scenarios, the results remain comparable to the centralized version if the network's data view encompasses a few neighboring gateways and the ED density is sufficiently high.

Furthermore, we implemented the MT and FD versions on the NebulaStream end-to-end processing platform, demonstrating the feasibility of a distributed implementation on such a system. To evaluate the performance improvements of the proposed variants in terms of execution times, we developed EXPLoSIM, a co-simulation framework that integrates a LoRaWAN network simulator with a Java execution environment for EXPLoRa-C threads. Extensive experimental results validate the effectiveness of the MT and FD versions under various assumptions about the computing and storage capabilities of Edge and Cloud nodes, as well as in heterogeneous network scenarios.

In future work, we plan to further refine the NebulaStream coordination logic to dynamically allocate computing tasks and data streams within the processing platform, adapting to changing network conditions and node availability. Additionally, we will explore the impact of our distributed SF allocation algorithm on reliability, particularly in scenarios where one or more gateways may go offline.

Moreover, this research aligns with our ongoing efforts in another project [26] focused on developing a fully distributed LoRaWAN network, where the network server's operations are decentralized and distributed among various gateways. In such a scenario, a distributed SF allocation algorithm, like the one we have presented here, becomes essential to maintain efficient and scalable network performance.

REFERENCES

[1] *Technical Characteristics for Low Power Wide Area Networks Chirp Spread Spectrum (LPWAN-CSS) Operating in the UHF Spectrum Below 1 GHz*, Standard TR 103 526 V1.1.1, 2018.

[2] (2018). *LPWAN Market Report 2018–2023*. [Online]. Available: https://iot-analytics.com/product/lpwan-market-report-2018-2023/

[3] D. Garlisi, A. Pagano, F. Giuliano, D. Croce, and I. Tinnirello, "A coexistence study of low-power wide-area networks based on LoRaWAN and sigfox," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Mar. 2023, pp. 1–7.

[4] LoRa Alliance Technical Committee. (2017). *LoRaWAN 1.1 Specification*. [Online]. Available: https://lora-alliance.org/resource-hub/lorawantm-specification-v11

[5] C. A. Hornbuckle, "Fractional-N synthesized chirp generator," U.S. Patent 7 415 791, Sep. 7, 2013.

[6] Y. Hajjaji, W. Boulila, I. R. Farah, I. Romdhani, and A. Hussain, "Big data and IoT-based applications in smart environments: A systematic review," *Comput. Sci. Rev.*, vol. 39, Feb. 2021, Art. no. 100318.

[7] D. Garlisi, I. Tinnirello, G. Bianchi, and F. Cuomo, "Capture aware sequential waterfilling for LoRaWAN adaptive data rate," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 2019–2033, Mar. 2021.

[8] S. Zeuch, X. Chatziliadis, A. Chaudhary, D. Giouroukis, P. M. Grulich, D. P. A. Nugroho, A. Ziehn, and V. Mark, "NebulaStream: Data management for the Internet of Things," *Datenbank-Spektrum*, vol. 22, no. 2, pp. 131–141, Jul. 2022.

[9] (2019). *LoRaWAN? Regional Parameters RP002-1.0.0*. [Online]. Available: https://lora-alliance.org/resource-hub/lorawanr-regional-parameters-rp002-100

[10] D.-Y. Kim, S. Kim, H. Hassan, and J. H. Park, "Adaptive data rate control in low power wide area networks for long range IoT services," *J. Comput. Sci.*, vol. 22, pp. 171–178, Sep. 2017.

[11] K. Q. Abdelfadeel, V. Cionca, and D. Pesch, "Fair adaptive data rate allocation and power control in LoRaWAN," in *Proc. IEEE 19th Int. Symp. World Wireless, Mobile Multimedia Networks*, Jun. 2018, pp. 14–15.

[12] S. Li, U. Raza, and A. Khan, "How agile is the adaptive data rate mechanism of LoRaWAN?" in *Proc. IEEE Global Commun. Conf.*, Dec. 2018, pp. 206–212.

[13] *The Things Network*. Accessed: Apr. 16, 2020. [Online]. Available: https://www.thethingsnetwork.org/docs/lorawan/adaptive-data-rate.html

[14] F. Cuomo, M. Campo, A. Caponi, G. Bianchi, G. Rossini, and P. Pisani, "EXPLoRa: Extending the performance of LoRA by suitable spreading factor allocations," in *Proc. IEEE 13th Int. Conf. Wireless Mobile Comput., Netw. Commun. (WiMob)*, Oct. 2017, pp. 1–8.

[15] B. Reynders, W. Meert, and S. Pollin, "Power and spreading factor control in low power wide area networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[16] G. P. Pinto, P. K. Donta, S. Dustdar, and C. Prazeres, "A systematic review on privacy-aware IoT personal data stores," *Sensors*, vol. 24, no. 7, p. 2197, Mar. 2024.

[17] H. Zhou, K. Jiang, S. He, G. Min, and J. Wu, "Distributed deep multi-agent reinforcement learning for cooperative edge caching in Internet-of-Vehicles," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 9595–9609, Jun. 2023.

[18] V. C. Pujol, B. Sedlak, P. K. Donta, and S. Dustdar, "On causality in distributed continuum systems," *IEEE Internet Comput.*, vol. 28, no. 2, pp. 57–64, Mar. 2024.

[19] A. Saleh, P. K. Donta, R. Morabito, N. H. Motlagh, and L. Lovén, "Follow-me AI: Energy-efficient user interaction with smart environments," 2024, *arXiv:2404.12486*.

[20] Q. Wang, B. Lee, N. Murray, and Y. Qiao, "MR-IoT: An information centric MapReduce framework for IoT," in *Proc. 15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*, Jan. 2018, pp. 1–6.

[21] F. Li, R. Xie, Z. Wang, L. Guo, J. Ye, P. Ma, and W. Song, "Online distributed IoT security monitoring with multidimensional streaming big data," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4387–4394, May 2020.

[22] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *Proc. 19th ACM Symp. Operating Syst. Princ.*, Oct. 2003, pp. 29–43, doi: 10.1145/945445.945450.

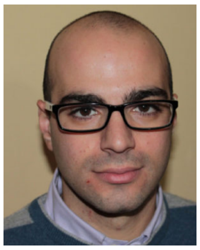
[23] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, p. 10.

[24] A. Alexandrov, R. Bergmann, S. Ewen, J.-C. Freytag, F. Hueske, A. Heise, O. Kao, M. Leich, U. Leser, V. Markl, F. Naumann, M. Peters, A. Rheinländer, M. J. Sax, S. Schelter, M. Höger, K. Tzoumas, and D. Warneke, "The stratosphere platform for big data analytics," *VLDB J.*, vol. 23, no. 6, pp. 939–964, Dec. 2014.

[25] *LoRaSim is a Discrete-Event Simulator Based on SimPy for Simulating Collisions in LoRa Networks and to Analyse Scalability.* [Online]. Available: http://www.lancaster.ac.uk/scc/sites/lora/

[26] P. Locatelli and F. Cuomo, "DeLoRaN: Decentralize LoRaWAN network server through blockchain," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2024, pp. 1–6.

PIETRO SPADACCINO received the degree (Hons.) in computer science engineering, in 2020, with a focus on the security and optimization of edge computing systems for the IoT, and the Ph.D. degree in information and communication technology (ICT), in 2024. His Ph.D. thesis is titled "Challenges and Opportunities in LoRaWAN Security: Exploring Protocol Vulnerabilities, Privacy Threats, and the Role of Edge Computing." Currently, he is an RTD-A Researcher with the Sapienza University of Rome, specializing in the optimization of edge computing systems.



DOMENICO GARLISI (Member, IEEE) has been a Visiting Researcher with the Department of Computer Science, UCLA. Since 2024, he has been with the National Context—Oriented Networking (CONLab) Laboratory, National Inter-University Consortium for Telecommunications (CNIT). He is currently an Assistant Professor with the University of Palermo, Italy. He has contributed to numerous research projects at both the national and international levels, leading two of them as the Principal Investigator. His research interests include wireless networks, software-defined radio, sensor networks, the IoT, and wireless systems for user localization and testing.

ANDREA FRANCESCHI received the bachelor's degree in computer engineering and the master's degree in engineering in computer science from the Sapienza University of Rome, in 2022. His master's thesis was in the field of IoT communications, with a focus on LoRaWAN technology and optimization of big data edge computing. Since then, he has been with the company Leonardo S.p.a.

ILENIA TINNIRELLO (Member, IEEE) received the Ph.D. degree in telecommunications engineering from the University of Palermo, in 2004. She was a Visiting Researcher with Seoul National University, South Korea, in 2004, and the Nanyang Technological University of Singapore, in 2006. She is currently a Full Professor with the University of Palermo. She has been involved in several European research projects. She is the Scientific Coordinator of European Digital Innovation Hub i-Nest. Her research interests include wireless networks, and in particular the design and prototyping of protocols and architectures for emerging reconfigurable wireless networks.

FRANCESCA CUOMO (Senior Member, IEEE) is currently a Full Professor with the Sapienza University of Rome, teaching courses in telecommunications, network infrastructures, and smart environments. She has advised numerous master's students in computer engineering and has been the advisor of 14 Ph.D. students in networking. She is the chair of the master's degree in data science in Sapienza. She has participated in several national and international research projects, being the Principal Investigator of many of them. She has authored over 185 peer-reviewed papers published in prominent international journals and conferences. Her Google Scholar H-index is 34 with more than 4700 citations. Her current research interests include low power wide area networks and the Internet of Things, 5G and open radio access networks, smart metering and vehicular networks, and multimedia networking.

Prof. Cuomo, she has been a part of the Scientific Committee of the Fondazione Ugo Bordoni and a Technical Member of Namex (Italian IXP), since May 2022. Relevant scientific international recognitions: two best paper awards. She has been the TPC Co-Chair of several editions of the ACM PE-WASUN Workshop and the 2016 ICCCN; the TPC Symposium Chair of IEEE WiMob 201; the General Co-Chair of the First Workshop on Sustainable Networking through Machine Learning and Internet of Things (SMILING), in conjunction with IEEE INFOCOM 2019; and the Workshop Co-Chair of AmI 2019: European Conference on Ambient Intelligence 2019. She will be the Track Chair of IEEE WCNC 2025. She has been on the editorial board of *Computer Networks* (Elsevier), *Ad-Hoc Networks* (Elsevier), IEEE TRANSACTIONS ON MOBILE COMPUTING, *Sensors* (MDPI), and *Frontiers in Communications and Networks* journal.

Open Access funding provided by ‘Università degli Studi di Roma "La Sapienza"’ within the CRUI CARE Agreement