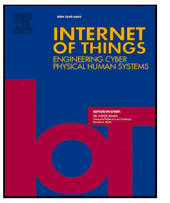
Internet of Things 27 (2024) 101266

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot



Research article

Edge2LoRa: Enabling edge computing on long-range wide-area Internet of Things

Stefano Milani [a], Domenico Garlisi [b,c], Carlo Carugno [a], Christian Tedesco [a], Ioannis Chatzigiannakis [a,c,*]

[a] Sapienza University of Rome, Rome, Italy
[b] University of Palermo, Palermo, Italy
[c] CNIT, National Inter-University Consortium for Telecommunications, Parma, Italy

ARTICLE INFO

Keywords:
Internet of Things (IoT)
Long Range (LoRa)
LoRa Wide-Area Network (LoRaWAN)
Low-Power Wide-Area Networks (LPWAN)
Edge computing
Edge-to-cloud continuum
Cloud–edge-terminal collaboration

ABSTRACT

Long-Power Wide Area Networks (LPWAN) is a low-cost solution to deploy very-large scale Internet of Things (IoT) infrastructures with minimal requirements following a classic producer/consumer model. Inevitably such deployments will require a shift towards low-latency, distributed and collaborative data aggregation models. The cloud edge computing continuum (CECC) has been proposed as an evolution of the traditional central ultra-high-end processing cloud into a continuum of collaborative processing elements distributed from the cloud to the network edge. Until today, incorporating existing centralized and monolithic LPWAN architectures in the CECC faces multiple security-related implications. We propose Edge2LoRa, a complete secure solution to incorporate LPWAN architectures in CECC enabling faster data processing while reducing the transmission of sensitive data. It improves network performance through data pre-processing, traffic flow optimization, and real-time local analysis. Edge2LoRa gradually transform existing LPWAN deployments into agile and versatile infrastructures that enable the seamless and efficient processing of data throughout the CECC while guaranteeing service continuity and full-backwards compatibility. We implement Edge2LoRa in hardware compliant with the Things Stack and the LoRaWAN v1.0.4 and v1.1. We evaluate the performance in terms of networking and computing resource utilization, quality of service and security. The results provide a clear indication of the improvements to public and private LoRaWAN infrastructures without any disruption or service degradation for existing legacy services. In public LoRaWAN deployments where large-scale IoT data streams drive big data analytics, we demonstrate core network bandwidth usage reductions of up to 90% and data processing latency improvements by a ×10 factor.

1. Introduction

The Internet of Things (IoT) has been a game changer in the way we operate, live, commute, and conduct business thanks to its capacity to monitor in real-time a broad range of environmental parameters in indoor, outdoor, industrial, urban, as well as rural areas [1]. From homes to cities, from farms to factories, in hospitals and vehicles, IoT deployments expand our ability to monitor physical phenomena and offer us the opportunity to significantly enhance the way we control our environment [2]. Our research

* Corresponding author.
E-mail addresses: milani@diag.uniroma1.it (S. Milani), domenico.garlisi@unipa.it (D. Garlisi), carugno.1999815@studenti.uniroma1.it (C. Carugno), tedesco.2025232@studenti.uniroma1.it (C. Tedesco), ichatz@diag.uniroma1.it (I. Chatzigiannakis).

https://doi.org/10.1016/j.iot.2024.101266
Received 16 February 2024; Received in revised form 16 June 2024; Accepted 23 June 2024
Available online 6 July 2024

focuses on the expected significant growth in the IoT sector, with IoT device connections predicted to rise from 7 billion in 2018 to 22 billion by 2025.

The proliferation of the IoT applications has significantly increased data generation across various fields such as informatics, urban planning, and economics. The International Data Corporation (IDC) predicts that the global data volume will reach a staggering 175 zettabytes by 2025. This explosion of data has created a growing need for the analysis of vast and diverse datasets produced by a multitude of IoT devices, often referred to as IoT big data. Within this diverse landscape of massive IoT use cases, emerging Low-Power Wide-Area Network (LPWAN) technologies introduce a uniform approach for creating global networks of devices for power-efficient and cost-effective integration of IoT data streams with cutting-edge web services across numerous specialized sectors [3–5].

In this work, we focus on Long-Range Wide-Area Network (LoRaWAN) as one of the most adopted LPWAN technologies [6,7]. LoRaWAN introduces a centralized, monolithic, architecture where network gateways (GWs) bridge traffic arriving from the IoT devices with the network backbone by directly forwarding all the IoT traffic to central cloud services [8]. Following a licence-free ad-hoc deployment model, it provides an ideal solution for connecting a wide range of IoT devices with minimal infrastructure requirements [9]. At the same time, the standard producer/consumer model adopted by LoRaWAN allows the straight-forward deployment of cloud-based analytics services to process the large amounts of IoT generated data streams [10]. For a graphical representation of the key components of LoRaWAN, see Fig. 1.

LoRaWAN technology is recognized for its suitability in supporting very large-scale deployments and managing substantial data volumes. The ETSI TR 103 526 (ETSI-EN-300) [11] document underscores the scalability and wideband capabilities of LPWAN LoRaWAN technology, particularly its unique ‘‘network densification’’ feature. The capture effect, an essential part of LoRa modulation, aids in multi-gateway setups by allowing at least one gateway to decode a packet successfully, even amidst collisions. Increasing gateway density can enhance data collection, improve the capture effect, and balance overall airtime. Another approach is the potential use of LoRa 2.4 GHz while reduces the range compared to traditional LoRa, it significantly enhances network capacity by enabling higher data rates and eliminating the need to comply with any duty cycle restrictions [12]. Specifically, the data rates for LoRa range from 0.3 kbit/s to 50 kbit/s depending on the spreading factor (SF) used [13], whereas LoRa 2.4 GHz supports data rates ranging from 0.595 kbit/s to 253.91 kbit/s. This higher data rate allows LoRa 2.4 GHz to facilitate the use of LoRaWAN for applications requiring lower latency, higher data rates, and non-scheduled packet transmissions. Incorporating CECC in such applications is crucial for meeting the demands of real-time operations over LoRaWAN.

Although LoRaWAN has served as an excellent starting point to integrate IoT infrastructures with cloud services and big data analytics, the centralized architecture faces certain bottlenecks when scaling up to large deployment areas. Increasing the number of gateways that forward large volumes of unprocessed IoT data to the centralized infrastructure places significant pressure on the network backbone in terms of bandwidth, energy and security [14]. At the same time, the emergence of new services that rely on new deployment and operational models require a shift from the classical two-tier producer/consumer model to multi-tier models involving client or application consumers, applications, and data sources. Such kinds of models necessitate a paradigm shift in the processing of IoT data for low-latency, distributed and collaborative aggregation. We are now entering a new phase where the fast and low-cost deployment strategy needs to be transformed into a scalable and agile approach that comprehensively considers these service requirements and optimally addresses them to enhance efficiency, sustainability, safety, and resilience.

Cloud Edge Computing Continuum (CECC) offers a natural evolution of information technology provisioning of computation and storage, which was traditionally bound to centralized data centers, to include resources available at the edges of the network [15,16]. In the case of LoRaWAN that adopts the design approach of using simple protocols to realize a centralized architecture, one faces multiple implications when trying to incorporate it in the CECC. Forcing the GWs to act as simple bridges, at one hand accommodates the rapid and low-cost deployment of unlicensed LPWANs, on the other hand excludes them from potentially acting as trusted intermediate processing and storage elements in the CECC. Recently some researchers have explored different approaches in modifying the specified operation of the protocols, proposing alternative architectures to reduce the significant pressure imposed on the central cloud services in cases of massive IoT data streams or time-constrained consumption of IoT data [17–19]. At the same time, introducing changes to the architecture and the specified operation of the protocols while maintaining backward compatibility is a challenging discourse [20]. As a result, to the best of our knowledge, there is currently no proposal that incorporates LoRaWAN in the CECC both for network management and application data processing that maintains the practical philosophy of LoRaWAN and respects backward compatibility.

We here propose a new approach that enables for the first time the efficient and secure integration of LoRaWAN in the CECC while respecting the current specifications and maintaining backward compatibility, thus allowing seamless interoperability between legacy and new elements. In other words, we allow the co-existence of cloud-centric analytics services with the execution of novel services that build upon edge computing concepts that require certain *Quality of Service (QoS) guarantees*. Consider that in the common case where IoT applications rely over public LoRaWAN infrastructures, like for example the TheThingsNetwork,[1] it is impossible to expect that the operator will support extended versions of the network server (NS) and/or the join server (JS) that deviate from the current standards and may lead to service disruptions for already deployed devices and services. For this reason, we guarantee that our extensions can be fully integrated in public LoRaWAN infrastructures: CECC-native GWs can co-exist with legacy GWs and communicate with a legacy network server (NS) and a legacy join server (JS) by respecting the protocols defined in the current LoRaWAN specifications. We also guarantee that the legacy IoT devices already deployed in the public LoRaWAN

[1] https://www.thethingsnetwork.org/

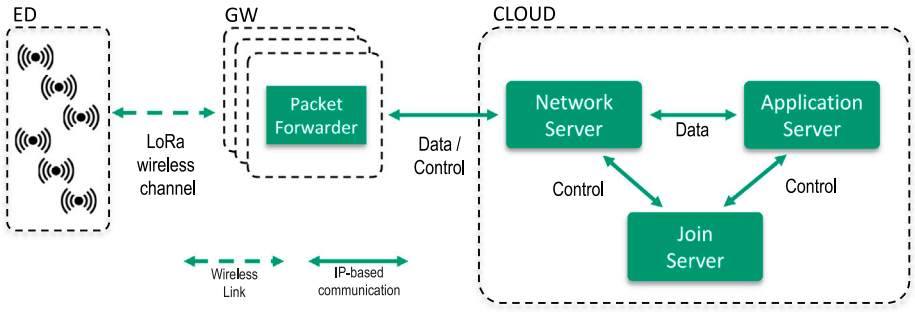


Fig. 1. Architecture of the LoRaWAN v1.0.4 and v1.1 Network Standards [8]. The figure illustrates the interrelations between the five key components: the end-device (ED), the gateways (GW), the network server (NS), the join server (JS) and the application server (AS). The GWs bridge traffic between the EDs and the NS without accessing the frame payload. The NS manages the network operations, the JS provides authentication, authorization, and access control to the EDs while IoT data streams are processed centrally by the AS.

infrastructure will not experience any service degradation when CECC-native elements are deployed. The extended infrastructure of legacy and CECC-native GWs can support multiple IoT applications, allowing the cooperative allocation of processing and storage of CECC-native resources. Only for the special case where an IoT use case requires that multiple QoS guarantees are respected concurrently, e.g., involving notions of exactly-once and at-most-once processing of messages, we require the deployment of an NS with extended functionalities. We remark, that the proposed solution is agnostic to the physical layer because it operates at the transport layer, similar to LoRaWAN. This flexibility allows it to seamlessly integrate with both the LoRa physical layer technologies without requiring modifications, thereby providing a versatile and robust framework for diverse IoT applications.

The main contributions of this paper can be summarized as follows:

1. We introduce Edge2LoRa, a new approach to transform the LoRaWAN GWs into an active element in the CECC by providing computational and storage resources at the edges of the network. The Edge2LoRa elements are designed in a way such that they respect the existing LoRaWAN specification and guarantee backward compatibility, allowing seamless interoperability between legacy and CECC-native elements, as demonstrated in the detailed performance evaluation included in the paper.
2. A mechanism that enables the association of IoT end-devices (ED) that are CECC-native to a specific CECC-native GW that will serve as the intermediate point for processing and storage in the CECC. The mechanism allows different strategies to be used for the association of GWs to EDs based on diverse optimization criteria of the network operation, the hardware capabilities and/or application-level metrics. Our design allows the *dynamic association* of EDs to GWs so that current network conditions and application-level parameters can be taken into consideration to optimize the available network, processing and storage resources.
3. An extended rejoin service procedure that allows the CECC-native ED, the associated GW and the application server (AS) to establish a group key. The group key is used by the CECC-native EDs to encrypt the IoT data before transmitting them to the LoRaWAN. Since the associated GW participates in the key generation phase, this mechanism enables the CECC-native GW to access the application payload of the LoRaWAN frames in a secure and privacy-preserving manner. This extended rejoin service is fundamental to allow the GWs to act as storage and processing elements within the CECC. Simultaneously, given that the group key is established after the deployment of the EDs in the area of interest, we avoid hard-coding the keys into the ED firmware and the GW software, ensuring they are not fixed for the entire duration of the network deployment.
4. A mechanism that provides certain *Quality of Service (QoS) guarantees* for the processing and storage of messages within the CECC even in the case of multiple deliveries of frames via legacy equipment. We provide detailed information on the operation of the network components and how they can co-exist with public infrastructures that rely on legacy NS and JS compliant with the LoRaWAN v1.0.4 or v.1.1 specifications.
5. A complete implementation of the proposed mechanisms based on the LoRaWAN specification v1.0.4 and v1.1 that is available as open-source software. The implementation allows third-parties to develop novel services that utilize all the resources available in the Cloud–Edge–Device processing and storage continuum. In the case when the new service relies on only one type of QoS guarantee for the processing and storage of messages, the service provider can deploy the CECC-native ED and GWs in existing public LoRaWAN infrastructures, e.g., such as the TheThingsNetwork. In the case when the new service relies on multiple types of QoS guarantees, the service provider will need to make sure that the NS incorporates an additional software component.
6. We conduct a multifaceted performance evaluation to assess the correctness, security and network utilization of the new system. We deploy our services in real-world devices in combination with an IoT device emulator to recreate massive IoT application scenarios. We also deploy a novel service by deploying CECC-native EDs and GWs in an existing public LoRaWAN infrastructure to demonstrate that legacy and CECC-native services can co-exist without any disruption or performance degradation.

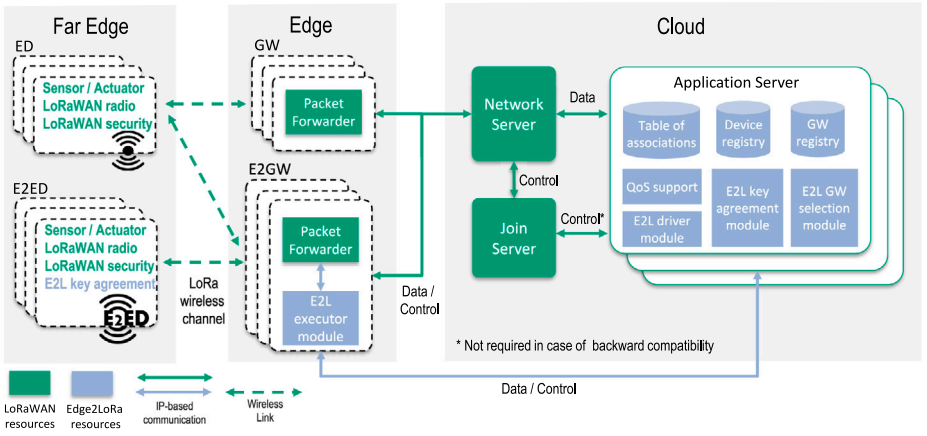


Fig. 2. Overview of the proposed Edge2LoRa (E2L) architecture including the key components of the current LoRaWAN network standard along with the ones introduced in Edge2LoRa and their interconnections. The diagram depicts the data flow for frames transmitted by an Edge2LoRa end device (E2ED) that is received by an Edge2LoRa GW (E2GW), which can be processed locally before forwarding through the Edge2LoRa driver to the AS (blue data link). Concurrently, frames transmitted by an E2ED and received by a legacy GW, or frames transmitted by a legacy ED and received by an E2GW, are forwarded to the NS (green data link), following the current LoRaWAN network standard.

The rest of this paper is organized as follows. Section 2 proposes the design of the improved LoRaWAN communication protocol and provides technical details on the key components. Specifically, Section 2.3.1 presents key elements of the current LoRaWAN standard with particular attention to the security aspects. An overview of the implementation of the proposed system along with some insights on the interconnection of the components is provided in Section 3. A detailed performance evaluation of the proposed approach is presented in Section 4. Section 5 summarizes very recent, relevant results and finally, Section 6 concludes the paper.

2. Enabling edge computing in LoRaWAN

The Edge2LoRa solution evolves the LoRaWAN architecture by introducing several elements: the device and GW registries, the table of device–gateway associations, the GW selection module, the group key agreement module, the edge processing executor module, the QoS support mechanism and the driver module. Fig. 2 depicts how these newly introduced elements interconnect with the components currently defined in the LoRaWAN network standard. Notice that the new components that constitute the Edge2LoRa architecture are depicted in blue color, while the components already defined by the LoRaWAN standard are depicted in green color. Remark that the standard components, i.e. those in green color, are also in Fig. 1.

The *far-edge layer* comprises legacy LoRaWAN IoT EDs and EDs that are compatible with the Edge2LoRa framework, denoted as E2ED. At *the edge layer*, the legacy LoRaWAN GWs provide the *Packet Forwarder module*, as defined in the LoRaWAN network standard, along with GWs compatible with the Edge2LoRa framework, denoted as E2GW. The E2GW introduces the Edge2LoRa executor, a lightweight and thin application virtualization layer for the execution of container-based micro-services and/or Big Data analytics operators on the received sensor data streams. Finally, at *the cloud layer*, each AS maintains the device registry, the GW registry and the table of device–gateway associations. These are in constant communication with the NS and JS so that the GW selection module can make Pareto-optimal decisions based on network performance, analytics execution performance, energy efficiency, security, reliability and dependability. The key agreement module establishes a secure communication channel between the E2ED, the E2GW, and the AS. The Edge2LoRa driver is responsible for the facilitation of the execution flow of Big Data operators on IoT data streams and/or the migration of container-based micro-services to the E2GWs. Finally, the QoS support module ensures that edge-based processing provides certain guarantees.

In the overall system that combines legacy and Edge2LoRa elements, frames transmitted by an E2ED that are received by a E2GW can be processed at the edge layer before they are forwarded to the Edge2LoRa driver residing at the AS. In Fig. 2 this execution and data flow is depicted using the blue data path. On the other hand, frames transmitted by an E2ED that are received by a legacy GW and/or frames transmitted by a legacy ED that are received by an E2GW will be directly forwarded to the NS (green data link), without any processing at the edge layer, as per the current LoRaWAN network standard. Since LoRaWAN frames may be received by multiple GWs, some of them might be processed at the edge, while duplicates may arrive unprocessed at the AS. Therefore correct distributed processing of multiple sensor data stream processing tasks may require specific QoS guarantees. It is up to Edge2LoRa driver in coordination with the NS to ensure the correct processing of the data streams that respect different QoS policies, including *"at most once"*, *"at least once"* and *"exactly once"* notions to ensure appropriate data management in different situations. For more details on the QoS guarantees see Section 2.4.

Typically frames in LoRaWAN contain *data* of the collected or processed information of a physical phenomenon as observed by an IoT device. In the standard LoRaWAN network standard, the ED encrypts the data included in the payload of a frame using a common key established between the ED and the AS. Therefore in the current LoRaWAN standard, neither the GW nor the NS are

capable of decrypting the data part of the payload of the frames received. In Edge2LoRa, access to the data included in the payload of the frames is enabled by establishing a group key between the Edge2LoRa driver included in the AS, the E2GW and the E2ED. The group key agreement protocol introduced in this paper relies on light-weight asymmetric encryption techniques that ensure the confidentiality of the exchanged data between E2ED, E2GW, and the Edge2LoRa driver, while at the same time keeping at minimum the energy and memory consumption. For more details see Section 2.3.

In the following sections we look into the modules that make up Edge2LoRa and how they operate throughout the lifecycle of the network, from the deployment of the gateways and devices to the actual operation and data processing.

2.1. Gateway and device deployment in Edge2LoRa

The deployment of an Edge2LoRa GW (E2GW) in a LoRaWAN requires to follow the GW registration process defined in the current LoRaWAN network standard: the 64-bit globally unique identifier of the GW (*GatewayEUI*) is provided to the NS and it is also inserted in the configuration of the legacy packet forwarder module of the GW along with the hostname or IP address of the NS. After this standard step, the *GatewayEUI* is also inserted in the *GW registry* of Edge2LoRa along with information regarding the processing and storage capabilities of the E2GW as well as the capabilities of the IP-level network in terms of, e.g., available bandwidth and latency. Naturally, the network level parameters are monitored continuously to reflect the current conditions.

Similarly, the deployment of an Edge2LoRa device (E2ED) requires first to follow the OTAA join process flow defined in the current LoRaWAN network standard. First, the 64-bit globally unique identifier of the device (*DevEUI*) along with the 64-bit globally unique identifier of the network the device is joining (*JoinEUI*) needs to be hardcoded in the E2ED firmware along with the necessary information to setup the end-to-end encryption between the E2ED and the NS and AS. Second, these information are stored in the JS that is overseeing the standard LoRaWAN join procedure. Third, these information are also inserted in the *device registry* of Edge2LoRa. In addition, the AS may introduce application-level metrics for each ED. These metrics are also stored in the device registry.

2.2. Gateway and device association in Edge2LoRa

In the current LoRaWAN network standard, the GWs are acting as bridges that directly forward the packets received from the wireless medium to the NS through the IP-based network. Recall that the GWs are incapable of decrypting the payload of the frames received. Moreover, due to the license-free ad-hoc deployment model adopted, gateways may have overlapping areas of network coverage (a) resulting in an increase of network traffic at the network backbone as frames are relayed to the NS multiple times, (b) may cause unexpected frame collisions and duty-cycle exhaustion.

On the other hand in Edge2LoRa the E2ED is associated with one E2GW that is enabled to process the data included in the payloads of the frames transmitted by the E2ED. The association of devices with GWs is carried out centrally based on the application specific needs. Such an approach has several benefits.

First, the radio-level statistics received by the NS from the GWs are used for optimizing data rates, airtime and energy consumption in the network for each device individually [8]. Moreover, a global view of the network-level and radio-level performance allows to reach Pareto-optimal assignments that (i) optimize power consumption of the device while ensuring that frames are still received at the associated E2GW; (ii) equalize the Time-on-Air of frames transmitted by the E2ED in each spreading factor's group; (iii) balances the assignment of E2ED to E2GW and the use of spreading factors across multiple E2GW and (iv) keep into account the channel capture [21].

Second, IoT deployments consist of very diverse devices which require system support for a wide range of resource capabilities and security requirements. Some services may require low-latency processing of IoT data utilizing specific hardware and software capabilities. Certain GWs may be small, battery operated, possibly mobile or deployed in outdoor environments potentially vulnerable to security threats while others may have access to specific heterogeneous hardware accelerators, such as Graphic Processing Units (GPUs) and Field Programmable Gate Arrays (FPGAs), thus achieving higher data processing throughput and energy-efficient execution. A centralized assignment allows a *hardware-conscious* optimization of the available resources in a holistic way [22].

Third, in cases when sensor data stream processing tasks are executed via a centralized Big Data processing framework, the operators of each task are organized in a data-flow graph that models the dependency between the operators. Operators may split data streams into windows of finite size based on criteria like time intervals, element counts, or sliding windows over which the computations are applied. In other cases join operators may combine frames arriving from the same E2ED or frames generated by multiple E2EDs based on common attributes. Ideally, each operator should be executed at the most suitable point across the CECC. Suitability is related to the available processing, storage, energy and communication resources and data access. Big Data frameworks have as an ultimate goal to scale out execution to increase performance by employing elaborate *scheduling* algorithms are used to identify the optimal execution plans that distribute data across the CECC. Information of such optimal execution plans will enable decisions such as where to execute each operator allowing intelligent resource selection and allocation [23].

The above dimensions provide a broad range of design choices for an optimization assignment of E2ED-E2GW. An example Pareto-optimal optimization flow diagram is illustrated in Fig. 3. The example indicates several dimensions taken into account along with their respective architectural collection points.

The table of device–gatewey associations provides a comprehensive view of the network topology by storing one entry for each device–gateway pair based on the radio-level statistics provided by the NS. In particular, during the operation of the

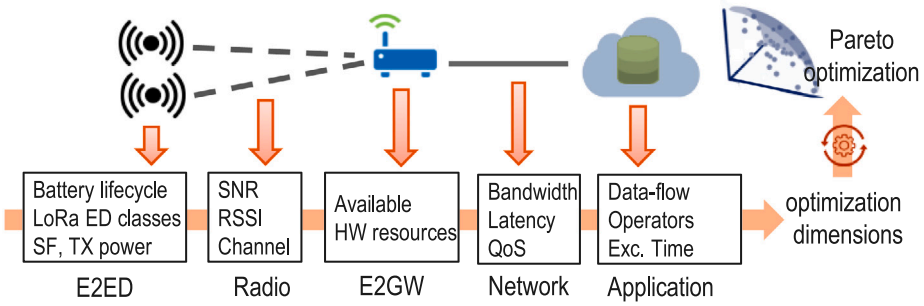


Fig. 3. Example of a Pareto optimization strategy considered in Edge2LoRa.

network, as defined in the current LoRaWAN standard, when a GW receives a frame from an ED it carries out certain link-level budget/measurements, such as the Received Signal Strength Indicator (RSSI) and the Signal-to-Noise Ratio (SNR), and inserts them in the frame's header that is forwarded to the NS. The *table of device–gateway associations* of Edge2LoRa is continuously listening for such updates from the NS given the latest radio-level statistics received from the GWs. Upon receiving updated radio-level statistics, it updates the corresponding entry related to the specific device–gateway pair.

Since the registries and the table of associations are continuously updated Edge2LoRa can detect E2EDs and/or E2GWs that are no longer accessible, as well as changes in the topology due to the dynamic nature of the wireless medium, e.g., increase/decrease in noise producing interference, or due to changes of the device positions in case of passive or active mobility. Similarly, the IP-level network is continuously benchmarked in terms of available bandwidth and latency between E2GW-NS and E2GW-AS connections.

The GW selection module will associate each E2ED to an E2GW by consulting the device and gateway registries along with the *table of device–gateway associations* of Edge2LoRa. Remark that the specific detail of an assignment strategy is beyond the scope of this paper.

As soon as the assignment is done, the selected E2GW takes on the responsibility of coordinating uplink and downlink data traffic between the E2ED and the AS as well as facilitating local processing and storage of frames following the processing and/or storage tasks assigned to the E2GW. As a result, direct communication via an IP link is established between the E2GW and the AS. VPN channels are preferred for this purpose, as they follow established best practices for providing secure communication between NS and GWs in LoRaWAN infrastructure. Remark that although the E2GWs are connected to only one NS, the above process does not exclude the possibility that each E2GW is connected to one or more AS.

The final step is to provide a secure communication channel between the E2ED and the E2GW so that the latter can have access to payloads of the frames transmitted by the E2GW. This is done using the group key agreement protocol presented in the following section.

2.3. Establishing secure communication between device–gateway-application

As previously mentioned, we have designed a key agreement protocol to establish a secure communication channel among E2ED, E2GW, and AS. In this subsection, we will first provide an overview of the current standard. Following that, we will introduce our key agreement protocol definition.

2.3.1. Security background of LoRaWAN v1.1 standard

The LoRaWAN v1.1 standard supports two methods for registering and activating EDs on the network: Over-the-Air Activation (OTAA) and Activation by Personalization (ABP), employing a secure communication protocol with encryption and authentication to ensure data privacy and network security [24]. In this study we consider OTAA since it is more secure than ABP [25,26].

OTAA relies on the following information encoded on the firmware of the EDs: (i) *DevEUI*: a 64-bit globally unique identifier of the device; (ii) *JoinEUI*: a 64-bit globally unique identifier of the network the device is joining; (iii) Network Key (*NwkKey*): a 128-bit key; and (iv) Application Key *AppKey*: a 128-bit key. This information is used to generate the session keys that facilitate end-to-end encryption between an ED and the AS. Additionally, a frame counter safeguards the integrity of application data: the NS can detect potential spoofing attempts by examining the up-link counter values. Lastly, message integrity is maintained through the use of the Message Integrity Code (MIC), which serves as a cryptographic signature calculated using the AES-CMAC algorithm. The complete process is depicted in Fig. 4.

In more detail, initially the ED and the JS compute two keys that they will use during the OTAA method:

$JSEncKey = aes128\_enc(NwkKey, 0\times05|DevEUI|pad_{16})$

$JSIntKey = aes128\_enc(NwkKey, 0\times06|DevEUI|pad_{16})$

Afterwards, the final session keys are extracted as follows:

(1) **Join Request**: The ED sends a Join Request frame triggering the OTAA. It includes the *JoinEUI*, the *DevEUI* and the *DevNonce*, a counter incremented with every Join-request that helps prevent replay attacks. The Join Request frame is sent in clear, the MIC is computed using the *NwkKey*.

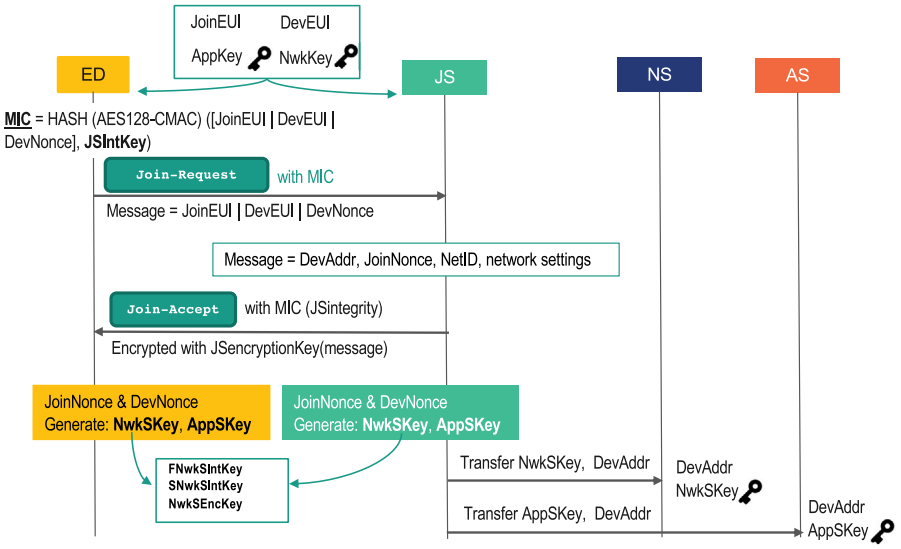


Fig. 4. LoRaWAN V1.1 join-procedure representation according to the Over-the-Air Activation (OTAA).

(2) **Join Accept**: Upon receiving the Join Request frame, the JS checks the integrity of the frame and the validity of the *DevNonce*. If successful, the JS assign a Device Address (*DevAddr*), a 32-bit identifier of the device unique inside the network and it generates a random 24-bit *JoinNonce*. The latter is used to prevent replay attacks and to ensure that the session keys are unique for each activation. It then sends a Join Accept message to the ED, which includes the JoinNonce, an identifier of the network, the DevAddr and some initial network configuration. The `JSEncKey` is used to crypt the JoinAccept frame, while the `JSIntKey` to compute and check the MIC of the JoinAccept and the Rejoin Request frames.

(3) **Key Generation**: The ED and the JS use the pre-shared information and the ones exchanged during the OTAA to compute three network keys and one application session key as follows:

– Forwarding Network Session Integrity Key (*FNwkSIntKey*) used to compute and check the Message Integrity Code (MIC) of the LoRaWAN up-link frames.

$$FNwkSIntKey = aes128\_encrypt(NwkKey, 0\times01|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

– Service Network Session Integrity Key (*SNwkSIntKey*) used to compute and check the MIC of the LoRaWAN downlink frames.

$$SNwkSIntKey = aes128\_encrypt(NwkKey, 0\times03|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

– Network Session Encryption Key (*NwkSEncKey*) used to crypt the network-level frames exchanged with the NS.

$$NwkSEncKey = aes128\_encrypt(NwkKey, 0\times04|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

– Application Session Encryption Key (*AppSEncKey*) used to crypt the application-level frames exchanged with the AS.

$$AppSEncKey = aes128\_encrypt(AppKey, 0\times02|JoinNonce|JoinEUI|DevNonce|pad_{16})$$

The process concludes with the JS transmitting the network session keys to the NS and the *AppSEncKey* to the AS.

Once activated, the ED can send and receive data by encrypting the payload using the *NwkSEncKey* or the *AppSEncKey* for network-specific and application-specific frames. The MIC is computed with the *FNwkSIntKey* or the *SNwkSIntKey* and appended to the frames before transmission. When the NS receives a frame, it checks the MIC using its copy of the *FNwkSIntKey*. If the MIC verification is successful, it ensures that the frame has not been tampered with and is from a legitimate source. Next, the encrypted payload is forwarded to the AS, which uses its copy of the *AppSEncKey* to decrypt the payload. The process is depicted in Fig. 5.

2.3.2. Group key establishment in Edge2LoRa

We here propose a group key agreement protocol that enables the creation of a shared session encryption key between the three involved actors. Remark that at this stage, the E2ED is already activated with the NS and the AS and the communication between

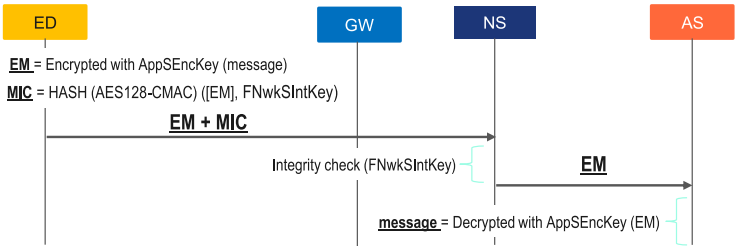


Fig. 5. Representation of the security protection through signed/symmetric encryption at network and application level.

Table 1
Proof of the Edge2LoRa group key agreement protocol.

E2ED	$EdgeSKey = Priv_{\text{E2ED}} \times G_{AS-\text{E2GW}} = Priv_{\text{E2ED}} \times (Priv_{AS} \times Pub_{\text{E2GW}}) = Priv_{\text{E2ED}} \times Priv_{AS} \times Priv_{\text{E2GW}} \times \mathcal{P}$	(1)
E2GW	$EdgeSKey = Priv_{\text{E2GW}} \times G_{AS-\text{E2ED}} = Priv_{\text{E2GW}} \times (Priv_{AS} \times Pub_{\text{E2ED}}) = Priv_{\text{E2GW}} \times Priv_{AS} \times Priv_{\text{E2ED}} \times \mathcal{P}$	(2)
AS	$EdgeSKey = Priv_{AS} \times G_{\text{E2GW}-\text{E2ED}} = Priv_{AS} \times (Priv_{\text{E2GW}} \times Pub_{\text{E2ED}}) = Priv_{AS} \times Priv_{\text{E2GW}} \times Priv_{\text{E2ED}} \times \mathcal{P}$	(3)
KEY	$Priv_{\text{E2ED}} \times Priv_{AS} \times Priv_{\text{E2GW}} \times \mathcal{P} = Priv_{\text{E2GW}} \times Priv_{AS} \times Priv_{\text{E2ED}} \times \mathcal{P} = Priv_{AS} \times Priv_{\text{E2GW}} \times Priv_{\text{E2ED}} \times \mathcal{P}$	(4)

the E2GW and the AS is secure. For simplicity we here assume that each E2ED is assigned to a single E2GW, however, the protocol can support any number of E2GWs.

Two shared session encryption keys are created between the E2ED, E2GW and the AS: the Edge Session Encryption Key (*EdgeSEncKey*) and the Edge Session Integrity Key (*EdgeSIntKey*). The former is used to enable secure encryption and decryption of the frame payload. The latter is used to check the integrity of the edge-specific frames. Here we propose the use of Elliptic Curve Cryptography (ECC), a public-key cryptography that uses elliptic curves over finite fields to create cryptographic keys [27,28]. ECC offers a higher level of security with smaller key sizes compared to other public-key cryptography techniques such as RSA [29]. This advantage enables the use of asymmetric encryption in resource-constrained scenarios, such as LoRaWAN EDs, with minimal energy consumption and memory consumption [25].

This approach helps to mitigate the computational overhead, particularly on the device, which is introduced by asymmetric cryptography compared to symmetric cryptography. Moreover, ECC, and hence asymmetric cryptography, is exploited only to compute the secret shared between the three parties. Once the two edge keys are computed, AES-128 bit encryption shall be used complying with the actual LoRaWAN specification.

The protocol is triggered by the E2ED, however, to optimize network utilization and reduce latency, the E2GW should share its ephemeral public ECC keys with the AS. Remark that the procedure is enforced on classical LoRaWAN application-specific frames where the payload is encrypted with *AppSEncKey* and integrity is performed with *FNwkSIntKey*. The E2GW and any other GWs in the radio range of the E2ED will forward the frame to the NS. The NS checks the integrity and, if successful, it sends the frame to the AS, discarding possible duplicates of the same frame. To simplify the presentation of the algorithm Algorithm 1 shows only the relevant steps for our proposal, not including steps of the LoRaWAN standard frame exchange.

Table 1 shows the proof that the three actors compute the same key. The E2ED (Eq. 1), the E2GW (Eq. 1), and the AS (Eq. 1) compute a value that is the multiplication of the three private ECC key and the same point P of the Elliptic Curve. Since the private ECC keys are scalar, the three actors have computed the same secret as shown in Eq. 1.

The *EdgeSKey* shall be used to derive the *EdgeSEncKey* and the *EdgeSIntKey* using a 128-bit hashing function as shown in Algorithm 1. The hashing function allows for greater flexibility in choosing the size of the elliptic curve used in the protocol, which impacts the size of the secret. This ensures that the security of the protocol is not compromised due to a constraint on the size of the curve.

The AS shall securely share only the *EdgeSIntKey* with the NS, to check the frame integrity in case the E2GW is not available, or the frame coming from legacy GWs.

Finally, the complete network lifecycle from LoRaWAN device activation to Edge2LoRa Cloud–Edge–Device coordination establishment is depicted in Fig. 6. The figure also includes the E2GW selection process discussed in the previous section.

It is worth noting that the E2GW remains active for legacy EDs, thus frames received from legacy EDs are still forwarded to the NS following the LoRaWAN standard.

2.4. Multiple deliveries and QoS support

In the current LoRaWAN standard the NS is responsible for identifying and removing frames received multiple times, selecting those with the best signal quality to forward to the AS. On the other hand, in Edge2LoRa deployment, multiple alternatives of frame routing, processing and storage may happen concurrently either at the edge or at the cloud. Frames consumed at an associated E2GW are not forwarded to the NS but instead the AS is notified directly about the outcome of the edge-based processing and/or storage. At the same time, frames received from legacy GWs coexisting in the infrastructures arrive at the AS via the NS. It is therefore crucial to make sure that duplicate processing and/or storage of data is avoided. It is the role of the Edge2LoRa driver module to coordinate the flow of data in the CECC.

Algorithm 1 Edge2LoRa Group Key Agreement

Require: ECC Curve with point $\mathcal{P}$ common to every actor.

ACTOR: E2ED
upon event $< E2LGKA,\ Init >$ **do**
$Priv_{\text{E2ED}} = random\_bytes()$
$Pub_{\text{E2ED}} = Priv_{\text{E2ED}} \times \mathcal{P}$
$EdgeSEncKey = \perp$
$EdgeSIntKey = \perp$
trigger $< Receive,\ EdgeJoinRequest|Pub_{\text{E2ED}},\ AS >$
end upon event

upon event $< Receive,\ EdgeJoinAccept|G_{AS-\text{E2GW}} >$ **do**
$EdgeSKey = Priv_{\text{E2ED}} \times G_{AS-\text{E2GW}}$
$EdgeSEncKey = hash_{128}(0x01|EdgeSKey)$
$EdgeSIntKey = hash_{128}(0x02|EdgeSKey)$
end upon event

ACTOR: E2GW
upon event $< E2LGKA,\ Init >$ **do**
$Priv_{\text{E2GW}} = random\_bytes()$
$Pub_{\text{E2GW}} = Priv_{\text{E2GW}} \times \mathcal{P}$
$EdgeSEncKey = \perp$
$EdgeSIntKey = \perp$
trigger $< Receive,\ PubInfo|Pub_{\text{E2GW}},\ AS >$
end upon event

upon event $< Receive,\ PubInfo|(G_{AS-\text{E2ED}}, Pub_{\text{E2ED}}) >$ **do**
$G_{\text{E2GW-E2ED}} = Priv_{\text{E2GW}} \times Pub_{\text{E2ED}}$
trigger $< Receive,\ PubInfo|G_{\text{E2GW-E2ED}},\ AS >$
$EdgeSKey = Priv_{\text{E2GW}} \times G_{AS-\text{E2ED}}$
$EdgeSEncKey = hash_{128}(0x01|EdgeSKey)$
$EdgeSIntKey = hash_{128}(0x02|EdgeSKey)$
end upon event

ACTOR: AS
upon event $< E2LGKA,\ Init >$ **do**
$Priv_{AS} = random\_bytes()$
$Pub_{AS} = Priv_{AS} \times \mathcal{P}$
$EdgeSEncKey = \perp$
$EdgeSIntKey = \perp$
end upon event

upon event $< Receive,\ PubInfo|Pub_{\text{E2GW}} >$ **do**
$G_{AS-\text{E2GW}} = Priv_{AS} \times Pub_{\text{E2GW}}$
end upon event

upon event $< Receive,\ EdgeJoinRequest|Pub_{\text{E2ED}} >$ **do**
trigger $< Receive,\ EdgeJoinAccept|G_{AS-\text{E2GW}},\ \text{E2ED} >$
$G_{AS-\text{E2ED}} = Priv_{AS} \times Pub_{\text{E2ED}}$
trigger $< Receive,\ PubInfo|(G_{AS-\text{E2ED}}, Pub_{\text{E2ED}}),\ \text{E2GW} >$
end upon event

upon event $< Receive,\ PubInfo|G_{\text{E2GW-E2ED}} >$ **do**
$EdgeSKey = Priv_{AS} \times G_{\text{E2GW-E2ED}}$
$EdgeSEncKey = hash_{128}(0x01|EdgeSKey)$
$EdgeSIntKey = hash_{128}(0x02|EdgeSKey)$
trigger $< Receive,\ IntKey|EdgeSIntKey,\ NS >$
end upon event

ACTOR: NS
upon event $< E2LGKA,\ Init >$ **do**
$EdgeSIntKey = \perp$
end upon event

upon event $< Receive,\ IntKey|EdgeSIntKey >$ **do**
$EdgeSIntKey = EdgeSIntKey$
end upon event

In the following paragraphs we examine the six possible scenarios of up-link frame transmissions. We consider the first three scenarios simple, in the sense that processing and/or storage of frames takes place exclusively a single point in the CECC, e.g., either at the cloud or at the network edge. The other three scenarios constitute more complex cases where some frames may traverse the network by following alternative paths that involve legacy and Edge2LoRa elements. In these latter scenarios it is crucial to ensure that the frames are not processed and/or stored multiple times throughout the CECC when it is required to respect *"at-most-once" or "exactly-once" QoS guarantees*.

2.4.1. Case 1: Frames transmitted by a legacy ED are received by both legacy GWs and E2GWs

In the first case, frames transmitted by a legacy ED are received by both a legacy GW and an E2GW. In this case the E2GW operates in full compliance with the LoRaWAN network specifications. In other words, the frames along with all the radio-level information collected from all GWs are forwarded to the NS. The NS uniquely selects those with the best signal quality and delivers

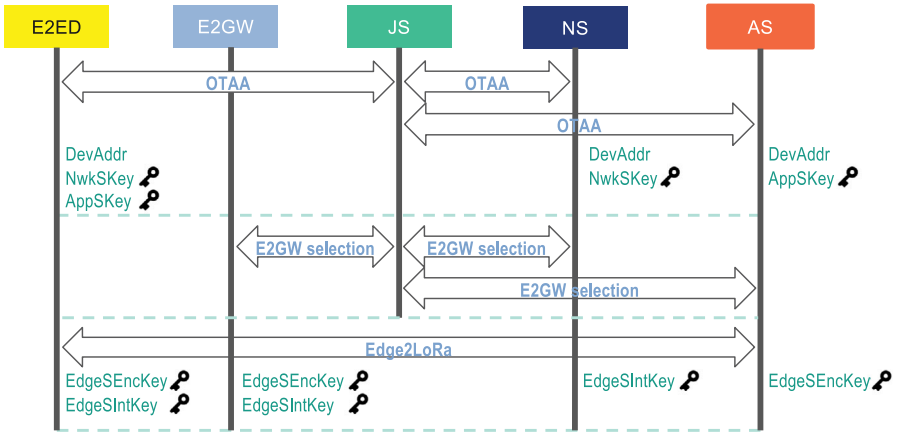


Fig. 6. LoRaWAN device activation and Edge2LoRa coordination procedures to enable edge processing while maintaining backwards compatibility.

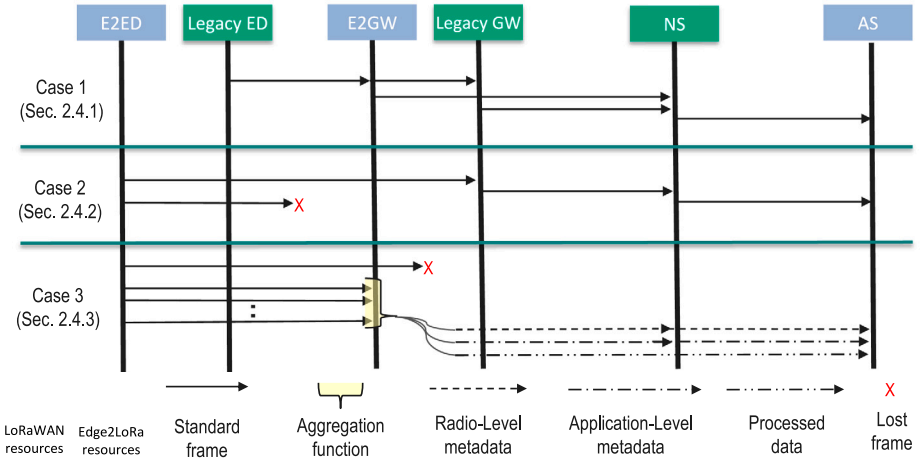


Fig. 7. Flow of frames in simple scenarios where legacy and Edge2LoRa elements co-exist in the IoT deployment. In cases 1 and 2 the network operates in full compliance with the LoRaWAN network specifications without being able to execute any processing and/or storage task in the CECC. Case 3 depicts the case where only Edge2LoRa elements are active and processing and storage tasks are executed at the network edge.

them to the AS while discarding any duplicate frames implementing the existing LoRaWAN protocols. Fig. 7 depicts this standard flow.

2.4.2. Case 2: Frames transmitted by a E2ED are received only by legacy GWs

The second scenario illustrates the case where the frames transmitted by an E2ED are received only by legacy GWs and thus no edge processing can take place. Like in the previous case, the frames are forwarded to the NS which will deliver them to the AS following the standard flow described in the LoRaWAN network specifications. Once again no processing and/or storage of frames can take place at the network edges. The flow is depicted in Fig. 7.

2.4.3. Case 3: Frames transmitted by an E2ED are received only by an E2GW

The third scenario considers one more simple case since the frames transmitted by an E2ED are received only by the associated E2GW without the activation of any legacy elements due to continuous failures. Such failures may involve permanent failures on the hardware equipment or continuous wireless interference blocking proper reception of the frames. This results in frames delivered to the AS via a single path. In this case the processing and storage of frames may be carried out exclusively at the associated E2GW. Fig. 7 depicts how the processing and/or storage of the data included in the payloads of the received frames is carried out at the network edge. The outcome of the edge-based processing and/or storage is forwarded directly to the Edge2LoRa driver module residing at the AS. At the same time, the collected radio-level information along with application-level metadata are forwarded to the NS so that the device and gateway registries are properly updated to reflect the current network conditions.

We should point out that the NS involvement is not always necessary for the processing of the application-level metadata. For instance, when considering backward compatibility with legacy NS the metadata only needs to reach the AS. The system will maintain its functionality because radio-level metadata will still be forwarded to the NS.

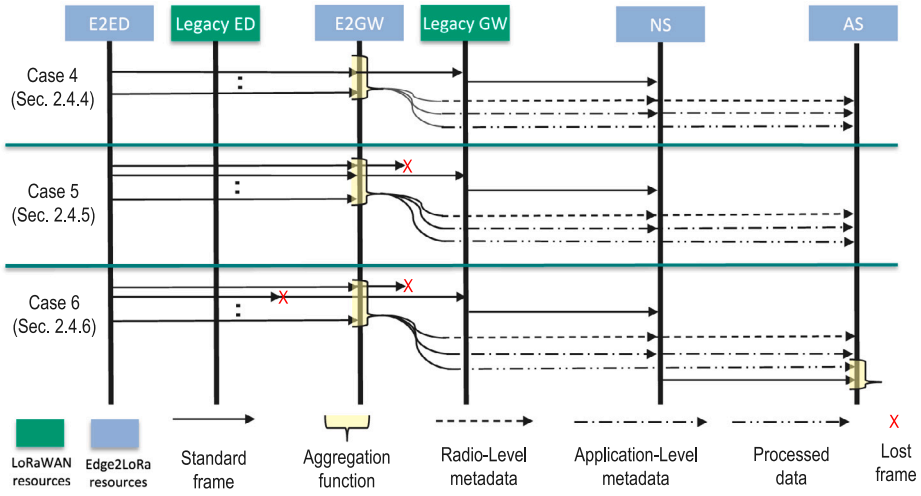


Fig. 8. Edge2LoRa frame flow in a mixed scenario through both legacy and Edge2LoRa elements. The figure also illustrates the flow of metadata involving the NS and the AS, taking into account modifications to the NS and the enabling of *"exactly-once"* QoS.

2.4.4. Case 4: Frames transmitted by an E2ED are received by both a legacy GW and by an E2GW

In the fourth scenario up-link frames arrive at the AS through multiple paths as shown in Fig. 8 without experiencing any transient of permanent failures. The frames received by legacy GWs are forwarded to the NS and are eventually delivered to the AS following the standard flow described in the LoRaWAN network specifications, similar to case 1. At the same time, the frames received by the associated E2GW may be processed and/or stored at the network edge, in a way similar to case 3. In this scenario we need to ensure that the frames are not processed and/or stored twice throughout the CECC when it is required to support *"at-most-once" or "exactly-once" QoS guarantees*. However we consider this to be a simple case since the scenario naively assumes that no transmission failures occur and that frames are always delivered through both paths. Under these assumptions, we can provide the necessary mechanisms for the AS to properly identify and discard duplicate frames without requiring any modification to the legacy NS.

In order to enable QoS by detecting processed frames at different levels, the Edge2LoRa protocol works to generate metadata used to notify processed frames together to the aggregation result. The Edge2LoRa driver module residing in the AS implements a duplicate detection filter (DDF) for identifying whether a given frame has previously appeared in a stream of data. The DDF is used to identify with no errors, duplicate frames in constant time [30]. The DDF relies on metadata produced by the E2GW upon receiving a frame that is about to be processed and transmitted to the AS. As shown in Fig. 8, there are two types of metadata generated: radio-level and application-level. The metadata are used to ensure that the device registries and table of associations are properly updated, like in case 3. At the same time application-level are also used to register the frame ID in the DDF maintained by the AS. In this way the AS will be notified about a frame being properly received at the E2GW and discard the one also delivered by the NS. After transmitting the metadata, the E2GW will process the frame as per the assigned processing and/or storage tasks. The outcome of the edge-based processing and/or storage may be transmitted to the AS at a later stage, as soon as the edge-based tasks are completed.

2.4.5. Case 5: All frames transmitted by an E2ED are received by an E2GW while some are also received by legacy GWs

The fifth scenario is similar to the fourth one, with the difference that now we assume that some frames are not received by a legacy GW. That is, we now assume a more realistic case where some transient failures occur either at the hardware level or during wireless transmissions however involving only the legacy GW. On the other hand no failures occur during the reception of the frames by the associated E2GW as depicted in Fig. 8.

Like in the fourth scenario, the DDF maintained by the Edge2LoRa driver module is enough to guarantee that the AS will be in a position to identify and discard the frames received by the legacy GW and arriving through the NS.

2.4.6. Case 6: Some frames transmitted by an E2ED are received only by a E2GW or only by an legacy GW

The sixth scenario represents the most realistic case where due to wireless transmission failures and/or transient failures at the hardware of both legacy and CECC-native hardware, some up-link frames arrive either through both paths or via one of the two paths as shown in Fig. 8. The complexity of this case is more evident if we consider that a processing and/or storage task may be composed by one or a combination of more operators that apply (i) *one-to-one and one-to-many transformations* of the received frames that are communicated directly to the AS and/or (ii) *many-to-one transformations* potentially aggregating data received from multiple frames where results are communicated after all frames received within a window are processed and/or stored.

When a processing and/or storage task relies on transformations that require the aggregation of multiple frames arriving over a window of time, we need to avoid inaccurate aggregations due to missing frames or frames being computed multiple times.

Table 2
Edge2LoRa module implementation details.

Component	Programming language	Principal used libraries
E2ED [31]	Arduino and RIOT-OS	Crypto (AES), micro-ecc (ECC)
E2GW	Rust	tonic (RPC), p256 (ECC), lorawan-encoding (Packet parsing, decryption, encryption)
AS	Python	Eclipse Paho (MQTT), gRPC (RPC), pycryptodome (Crypto)
ED emulator	NodeJS	lora-packet (Packet parsing, encryption)

We achieve this by assigning a timeout on the processing and/or storage of each single frame delivered to the AS via the NS. The duration of the timeout is appropriately set by the Edge2LoRa driver module in a way such that it will allow the E2GW to complete the processing of the operator. The Edge2LoRa driver module decides on the length of the timeout based on a series of factors taking into consideration the type and combination of the operators used along with the data-flow graph that models the dependency between the operators and the hardware-level, network-level and application-level operation parameters of the E2GW. The Edge2LoRa driver module relies on the up-to-date information stored in the device and gateway registries. For example, for filter operators the timeout can be set to zero; for map operators that carry out one-to-one or one-to-many transformations the timeout can be set in relation to the latency of the E2GW-AS link and the computational resources of the E2GW; for window operators, the timeout needs to also consider the size of the window.

Depending on the actual task and given the presence of frames being delivered via legacy elements, the AS may have to repeat the processing and/or storage tasks carried out by the E2GW. This will require the transmission of the raw data included in the frames received only by the E2GW to the AS so that the transformations can be accurately executed respecting the required QoS policies. *"At-most-once"* is the cheapest with the least implementation overhead and highest performance because it can be done in a fire-and-forget fashion without keeping the state in the E2GW or at the E2ED. Guaranteeing *"at-least-once"* requires multiple transmission attempts in order to counter transport losses which means keeping the state at the device and utilizing the LoRaWAN acknowledgment mechanism. The *"exactly-once"* is the most expensive and has consequently the worst performance because, in addition to *"at-least-once"* that relies on the LoRaWAN acknowledgment mechanism, it requires the state to be kept at the NS in order to filter duplicate deliveries. Additional implementation details are provided in Section 3.1.3.

3. Implementation details

3.1. The Edge2LoRa extensions

The implementation of the Edge2LoRa services depicted in Fig. 2 is available as open-source project[2] to freely download and execute in public or private LoRaWAN deployments. We here note that a previous version of the code has been demonstrated at MobiCom2023 [31]. This demonstration involved the use of five physical end devices (`Heltec Cubecell HTCC-AB01`[3] & `HTCC-AB02`[4]), two of which were implementing the legacy LoRaWAN components while three included also the Edge2LoRa extensions. The demonstration also included a NS based on the Things Stack,[5] compatible with the LoRaWAN 1.0.4 specification. The demonstration aimed to validate the system's functionality within private deployments in a scaled-down scenario.

Table 2 summarizes the complete implementation details for the different components deployed through the three layers of the architecture.

3.1.1. Far-edge layer

At the far-edge layer we implement the Edge2LoRa Key Agreement module based on the `secp256r1`[6] elliptic curve. We selected this curve as it employs 256-bit private keys and 512-bit public keys. Remark that the public keys can be compressed to 33 Bytes, making them suitable for transmission over a single LoRaWAN frame. The computed secret key is of the same size as the private key. Subsequently, it undergoes two SHA256 hashing processes, each concatenating the secret with a byte, differing in the two rounds. The first 16 bytes of the result are then used to generate the final *EdgeSEncKey* and *EdgeSIntKey*.

We provide a hardware-agnostic implementation of the Edge2LoRa Key Agreement module using the *RIOT Operating System* [32] since it supports many different architectures for 8 bit, 16 bit, 32 bit and 64 bit processors, provides a simple process manager with support for multi-threading, provides a generic network stack and also power management [33]. We also use *Arduino* to provide a hardware-specific implementation of the module for the `Heltec Cubecell HTCC-AB01 and HTCC-AB02` as they provide a native implementation of LoRaWAN 1.0.4.[7] Both RIOT OS and Arduino incorporate the `micro-ecc` library [34] that implement

[2] https://github.com/Edge2LoRa
[3] https://heltec.org/project/htcc-ab01-v2/
[4] https://heltec.org/project/htcc-ab02/
[5] https://github.com/TheThingsNetwork/lorawan-stack
[6] https://neuromancer.sk/std/secg/secp256r1
[7] https://github.com/HelTecAutomation/CubeCell-Arduino

ECDH and ECDSA for 8-bit, 32-bit, and 64-bit processors. The implementation of the 128-bits AES encryption was based on the `crypto` module provided by *RIOT OS* [35] and by *Arduino*.[8] These cryptographic functions can be used within security protocols at the system level by providing seamless crypto support across software and hardware components [36]. A detailed evaluation of the energy consumption overhead incurred by the Edge2LoRa Key Agreement module due to the use of the ECC on different common IoT chips is available in [25]. Moreover, this implementation can be easily replicated on other hardware/software platforms. This is because the LoRa driver, which is a key component in the system, is not modified. This means that the changes are largely agnostic to the underlying hardware and software, making them versatile and easy to implement across different systems.

3.1.2. Edge layer

At the Edge layer we implement the Edge2LoRa executor module that is deployed at the E2GW. The executor is interconnected with the standard *Packet Forwarder* of the Semtech[9] through a light-weight proxy component implemented in Rust. The proxy intercepts traffic from the *Packet Forwarder* and redirects it to the Edge2LoRa executor module after it has been decrypted by the Edge2LoRa Key Agreement module executed within the E2GW. The LoRaWAN packet parsing, decryption and encryption are implemented with the *lorawan-encoding* crate,[10] while the Elliptic Curve cryptographic primitives are implemented using the *p256* crate.[11] The communication over the RPC Protocol is implemented using the *tonic* create,[12] which is a production-ready implementation of *gRPC*[13] for Rust. The implementation of the actual Edge2LoRa executor module is not within the scope of this paper as several choices exist depending on the resources available on the E2GW.

3.1.3. Cloud layer

For the implementation of the Edge2LoRa extensions residing at the cloud layer we use the Thing Stack. It is implemented using Python3 and it interfaces with Thing Stack using the MQTT integration that the latter offers. Using the MQTT integration the Edge2LoRa components residing in the AS can receive the uplink LoRaWAN packets and schedule downlink ones. The MQTT interface is implemented with the Eclipse Paho library.[14] The communication between the AS and the E2GW exploit the RPC Protocol, implemented using the *gRPC Python3 SDK*.[15] All the cryptographic primitives are implemented using the *pycryptodome* library[16] which offers support for both symmetric and asymmetric cryptography and for hashing functions.

For the implementation of the different QoS policies, as mentioned in Section 2.4, modifications to the Network Server (NS) are not always required. This is particularly true when considering backward compatibility with legacy NS systems. The provision of the *"at-least-once"* and *"at-most-once"* QoS guarantees depends on the version of the LoRaWAN network specifications implemented by the NS. The main difference between the two versions of the specification lies in the check of frame integrity:

- **LoRaWAN 1.0.4**: No integrity check is computed by the NS. The NS checks the FCnt of the Edge2LoRa frame received, and if the check passes, it forwards it to the AS.
- **LoRaWAN 1.1.0**: The NS checks the frame integrity using the FNwkSIntKey. The MIC, used for this integrity check, is computed using the new EdgeSIntKey. Consequently, the integrity check of the NS will fail, resulting in the frame being dropped. Given that the NS will reject all Edge2LoRa frames, certain precautions must be taken by the E2ED or the E2GW to ensure the NS can effectively manage network parameters and downlink scheduling. Two potential solutions are proposed:
  (a) The E2ED periodically sends a legacy frame to the NS, allowing it to update the FCnt and schedule downlinks accordingly. While this solution does not compromise security, it mandates the E2ED to intermittently transmit a legacy frame, impacting the benefits of Edge Computing.
  (b) Using the secure channel established through the group key agreement, the E2ED shares the FNwkSIntKey with the E2GW. Subsequently, the E2GW can periodically generate simulated legacy frames containing the updated counters and an aggregation of the network metrics. These frames are then transmitted to the NS to facilitate updates. Although the latter solution does not compromise the advantages of Edge Computing, it introduces a potential security impact.
  The choice between these solutions involves a trade-off between security and uninterrupted edge-level operations.

Given these considerations, Edge2LoRa supports the *"at least once"* QoS guarantee in the case of LoRaWAN 1.0.4, while it supports the *"at most once"* QoS guarantee in the case of LoRaWAN 1.1.0, both fully supports the existing deployed structures, ensuring that the system's operational continuity is maintained. However, to support the *"exactly-once"* QoS guarantee the NS needs to support the DDF discussed above requiring the coordination between E2GW, the NS and the AS in combination with advance mechanisms for data buffering and retransmissions [37]. Table 3 summarizes the QoS guarantees that Edge2LoRa is capable of supporting, detailing all possible combinations of Edge2LoRa and legacy components.

[8] https://rweather.github.io/arduinolibs/crypto.html
[9] https://github.com/Lora-net
[10] https://docs.rs/lorawan-encoding/latest/lorawan_encoding
[11] https://docs.rs/p256/latest/p256/
[12] https://docs.rs/tonic/latest/tonic/
[13] https://grpc.io/
[14] https://eclipse.dev/paho/
[15] https://grpc.github.io/grpc/python/
[16] https://www.pycryptodome.org/

Table 3
QoS support for different configuration scenarios.

Network server	LoRaWAN Spec.	QoS	Backward compatibility	Edge2LoRa
Legacy	1.0.4	*at least once*	✓	✓
Legacy	1.1.0	*at most once*	✓	✓
Edge2LoRa support	1.0.4	*exactly once*	✗	✓
Edge2LoRa support	1.1.1	*exactly once*	✗	✓

3.2. Hybrid testbed for large scale experimentation

In contrast to the small-scale evaluation presented in [25,31] here we wish to look into large-scale LoRaWAN deployments. Moreover, in contrast to the previous evaluations, we are interested to evaluate the backwards compatibility of the provided Edge2LoRa extensions with the LoRaWAN network standard. It is therefore crucial to provide an environment that allows to experiment using a public infrastructure rather than a privately deployed one. For this reason, we developed a detailed emulator module for the End Devices, a tool designed to reproduce LoRaWAN frames received in a real-world network environment.

Our tool is a comprehensive emulator software of LoRaWAN EDs that allows the creation of extensive testing environments, encompassing thousands of EDs on a controlled network setup. The emulated EDs adhere to LoRaWAN v1.0.4 and v1.1 specifications and support the OTAA activation method. The modular architecture allows different types of LoRaWAN EDs (Class A, B, and C) to be modeled, each with the ability to send different payloads as defined by the configuration scripts. Emulated LoRaWAN EDs transmit their PhyPayload frames via UDP to the GWs, which encapsulate them into either Semtech UDP frame Forwarder (GWMP) messages for delivery to the NS, in case of legacy frame, or to the assigned E2GW executor according to the logic implemented in the Edge2LoRa driver.

The tool provides separate configuration parameters to control and fine-tune the activity of the legacy EDs and the E2ED. Parameters such as the EDs source rate and the message payload can be configured for each device individually. Moreover, the emulator can be connected to a real dataset of sensor values so that it can encapsulate the sensor values in the payloads of the transmitted frames, achieving in this way a higher fidelity. This feature allows for more accurate and realistic testing and evaluation of the system's performance under conditions that closely mimic real-world scenarios.

At the network level, real LoRaWAN network traffic is reproduced by incorporating the detailed frames loss model introduced in [21]. The interference model considers EDs deployed in geographic areas that are covered by multiple GWs. The coverage of each GW is represented by a circular radius. The model considers uniformly deploying EDs in areas where multiple GWs operate with possible overlapping coverage areas. Our emulator is fully aligned with the methodology described in the referenced paper [21] and is modeled by the following formula, reported as formula [21](13).

$$S_c(G_{sf}) = 2\pi \int_0^{R/\alpha} \delta_{sf} e^{-2\frac{\alpha^2 r^2}{R^2} G_{sf}} r \cdot dr + \\ +\delta_{sf}(\pi R^2 - \pi R^2/\alpha^2) e^{-2 \cdot G_{sf}}$$

The above formula generalizes the medium access control in LoRaWAN that follows Aloha scheme in a scenario where the EDs are uniformly distributed. The model is specifically formulated for a circular coverage area and the throughput is calculated in the presence of a channel capture effect. In the formula $\delta_{sf} = G_{sf}/(\pi R^2)$ is the density of traffic load offered to a SF=sf and R is the cell radius. Due to the scenario configuration, a target ED is actually competing with a fraction of devices generating the total load G_{sf}. Indeed, neglecting the effect of random fading and assuming an attenuation law of type $r^{-\eta}$, all the interfering nodes at a distance higher than $\alpha \cdot r$, with $\alpha = 10^{SIR/10\eta} > 1$, do not prevent the correct demodulation of the signal from the target ED. Here, SIR represents the Signal to Interference Ratio and η is the propagation coefficient. The smaller the α coefficient, the lower the number of competing EDs. Therefore, the throughput can be obtained by $S_c(G_{sf})/G_{sf}$. Finally, the model supports the presence of multiple GWs with overlapping areas, where M is the number of GWs deployed in the coverage area and $S_c(G)$ the throughput perceived under load G, the total capacity can be by approximated by $M \sum_{sf} S_c(G_{sf}/M)$.

Fig. 9 illustrates the testbed setup that we developed for the purposes of the large-scale performance evaluation. The ED layer and part of the GW layer are replaced by our emulator that communicates with software components executed in real-hardware elements. In the figure, light-blue blocks represent emulated components, while yellow blocks represent real deployment hardware elements. To validate the backward compatibility of the proposed architecture, we utilize the TheThingsNetwork public LoRaWAN infrastructure.

4. Performance evaluation

In this section, we present a comprehensive evaluation in terms of the network performance improvements and the security guarantees provided by the Edge2LoRa extensions to the current LoRaWAN network specifications. The results presented here provide insights on the advantages of our design choices and how they can be used to improve the operation of large-scale LoRaWAN deployments.

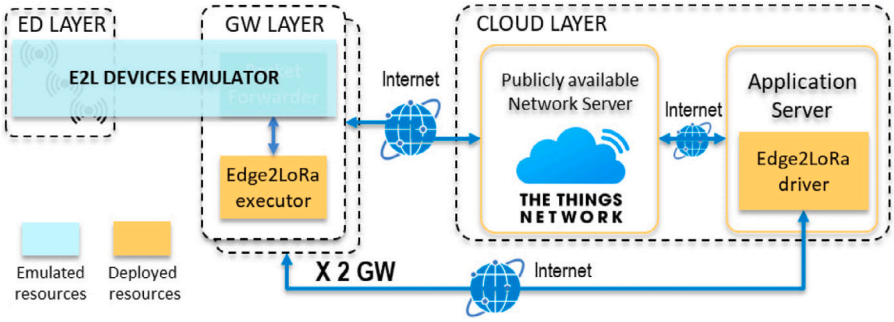


Fig. 9. Overview of the testbed that combines emulated elements with real-hardware communicating with the TheThingsNetwork, a public LoRaWAN infrastructure. The diagram is color-coded to distinguish between the emulated components (light-blue) and the actual hardware elements (yellow).

4.1. Edge2LoRa advantages in real-life scenarios

Several sectors can benefit from the capabilities of Edge2LoRa. For instance, in Water Metering and Flow Monitoring, edge processing improves computing latency. The Edge2LoRa solution enables faster data processing by leveraging the substantial computing capacity of network servers, thus enhancing the efficiency of water network monitoring [38]. Moreover, privacy preservation can benefit from edge processing as it reduces the transmission of sensitive data to cloud servers, thereby mitigating privacy risks and potential misuse or theft of user data. In the realm of water management, edge computing allows for the implementation of solutions to detect user anomaly patterns and identify water leakage, enhancing the overall security and efficiency of the water network [39].

Smart Buildings is another vertical application that can benefit from the Edge2LoRa solution that performs data pre-processing aggregation at IoT gateways, reducing network bandwidth consumption and addressing issues such as transmission delay and packet loss [40]. Furthermore, traffic flow control is enabled by migrating data processing and aggregation tasks to the edge, optimizing traffic flow, minimizing bandwidth requirements for end users, while maintaining data quality. Real-time local analysis is made possible by Edge2LoRa, enabling local data traffic analysis and real-time notifications to a local entity, facilitating efficient monitoring and management of smart building systems.

In the Smart Industry vertical, the enhanced security of the Edge2LoRa approach implements security measures to protect against attacks and data manipulation, ensuring the robustness of edge nodes, servers, and networks, thereby safeguarding critical industrial infrastructure. Finally, in Agricultural Applications, the solution takes into account power resource optimization [9]. It improves power resources and battery capacity in agricultural settings by incorporating a flexible task offloading scheme that considers the power resources of each device.

4.2. Network performance

In terms of network performance, first, we evaluate the performance of a large-scale LoRaWAN deployment that follows the standard network specification to serve as our baseline. Then we introduce the Edge2LoRa extensions and measure the improvements achieved under various traffic loads. Our goal is to highlight the benefits of the Edge2LoRa proposal and also demonstrate the backward compatibility of all the Edge2LoRa extensions when operated in a LoRaWAN infrastructure that is fully compliant with the current network standard.

4.2.1. Large-scale deployment scenario

In this paper, we focus on massive IoT scenarios that encompass dense deployments of a large number of IoT devices that require massive connectivity to efficiently transmit streams of sensor data to cloud services [3,4,6]. We look into deployments that are expected to reach device densities ranging from 1k to 10M devices per square kilometer [41]. Apart from the device density, other factors that need to be taken into consideration are the device duty-cycle, the sensor sampling rate and the message generation frequency. Furthermore, the density of the gateways and the resulting overlap in the network coverage needs to be taken into account since they affect the overall performance of the network [42].

We design a large-scale IoT network scenario where 3000 devices are uniformly deployed in a circular area of 1 km of radius, producing a total of 500 frames of 24 bytes each and using a transmission rate of 1 frame every 3 s. The LoRaWAN data rate is set to $DR = 5$ (spreading factor set to $SF = 7$ and bandwidth 125 KHz). Moreover, two GWs are deployed equally spaced at 0.15 km from the center of the circular area. The devices are progressively activated with an interval of 0.1 s, resulting in a transitory period of approximately 5 min before all the devices become active. The selected scenario produces a density of approximately 950 devices per square kilometer and generates 1000 frames per second. According to the formula (13) of [21], each GW perceives on average a network Frame Delivery Ratio (FDR) of 31%. The complete list of parameters used for the evaluation scenario is summarized in Table 4. For a graphical representation of the deployment scenario refer to Fig. 10.

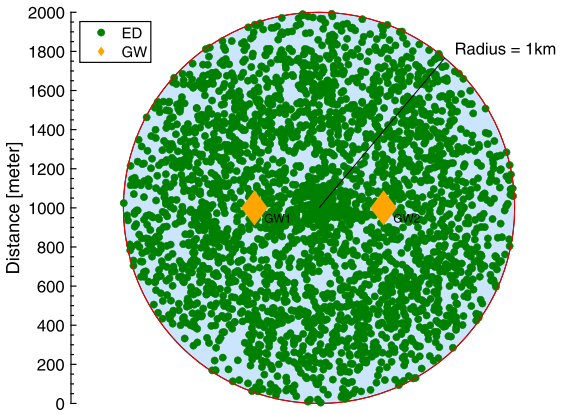


Fig. 10. A large-scale IoT network scenario comprised from 3000 EDs and 2 GWs uniformly deployed in a circular area of 1 km of radius.

Table 4
LoRaWAN scenario simulation parameters for the realistic large-scale scenario.

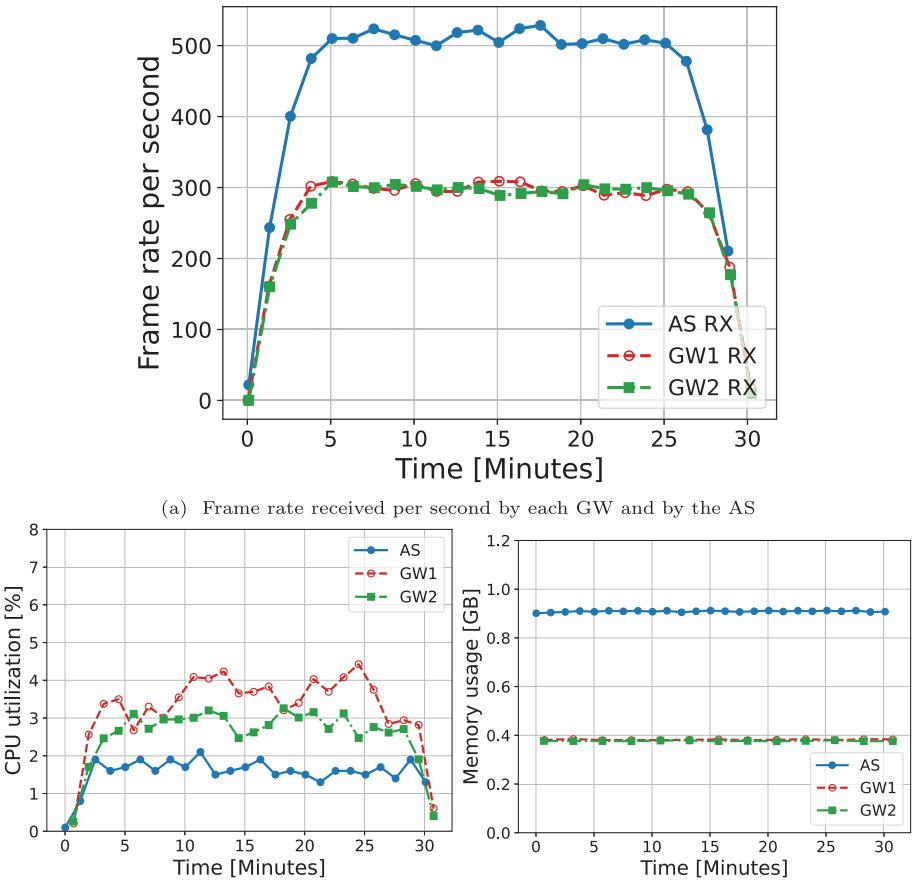
Parameter	Value
Deployment area	circular area of 1 km radius
Number of Gateways	2
Gateways deployment	equally spaced at 0.15 km from the center
Number of Devices	3000
Device deployment	uniform
Device activation pattern	progressively activated with an interval of 0.1 s
Device activation method	OTAA
Transmission Data Rate	$DR = 5$
Bandwidth	$BW = 125$ KHz
Spreading Factor	$SF = 7$
Propagation Coefficient	$\eta = 2.9$
Frames transmitted	500
Frame transmission rate	1 frame every 3 s
Frame size	24 bytes
Frame delivery ratio	based on interference model defined in [21]
Aggregation window size	30 s
Experiment duration	30 min

In the experiments presented here, the devices are configured to act as legacy ED or as E2ED. Similarly, the gateways are either set to legacy mode or E2GW. In the experiments that involve E2ED and E2GW, the Edge2LoRa driver is configured to assign half of the E2ED to one E2GW and the other half to the other E2GW. We use this simple approach of partitioning the E2ED to E2GW so that we can better understand the improvements in comparison with the performance of the legacy LoRaWAN components. The performance of different Pareto-optimal strategies, as those discussed in Section 2, are left as future work.

In terms of the high-level application, we introduce only one application where the edge-computing task deployed to the E2GW comprises of a series of one-to-one and many-to-one operators applied on a device-based window. We configure the window size to 30 s. As a result, we expect that for every 10 frames received from each E2ED the task will produce a single message containing the aggregated value of the 10 measurements included in the payloads of the original frames. Once again, other kinds of edge-computing tasks that reflect specific use-case scenario are left for future work. Since in LoRaWAN each device can belong to only one application, the gateways will execute the group key agreement module for each device only once. Therefore, the scalability of Edge2LoRa is independent of the number of applications but depends only on the number of devices. Moreover, since each device is assigned to only one gateway, there is no repetition of the execution of the group key establishment module across multiple number of gateways.

All experiments are carried out for a total duration of 30 min. We repeat each experiment multiple times until the certainty is above 95% to ensure the repeatability of our results.

In terms of hardware used throughout the experiments, for the gateways and E2GW we use Raspberry Pi 4 Model B units, each equipped with a Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.8 GHz and 4 GB of RAM running Debian GNU/Linux 11. The AS is deployed on an Intel NUC, supplied with an Intel i5-6260U CPU @ 1.80 GHz and 8 GB of RAM running Ubuntu 22.04.3 LTS. For the experiments that rely on the TheThingsNetwork public infrastructure, the NS is provided by the TheThingsNetwork. On the other hand, when we deploy a NS within our laboratory, we use an Intel NUC, similar to the one used to host the AS, that is an Intel i5-6260U CPU @ 1.80 GHz and 8 GB of RAM running Ubuntu 22.04.3 LTS.



(a) Frame rate received per second by each GW and by the AS

(b) CPU usage of the host machines accommodating the GW and the AS

(c) Memory usage of the host machines accommodating the GW and the AS

Fig. 11. Baseline performance for ED, GW and AS when only legacy components are activated via the TheThingsNetwork public infrastructure.

4.2.2. Performance metrics

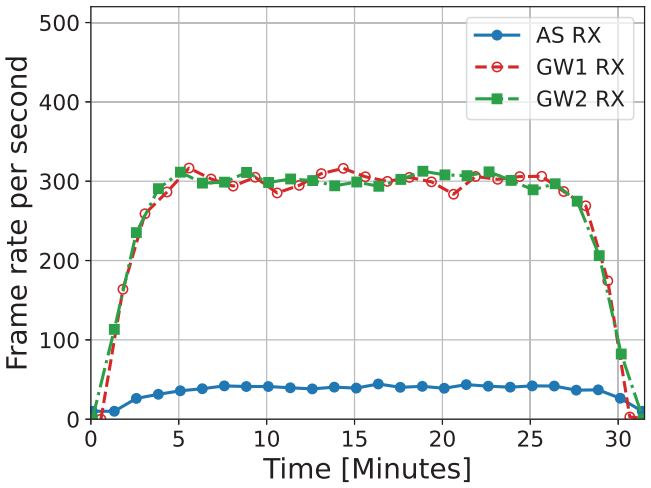
We have selected four metrics to evaluate the performance of the Edge2LoRa system: the total number of frames transmitted over the IP-based network, the number of frames delivered solely by one gateway, and the computational resources utilized by the host machine, specifically the percentage of CPU usage and the amount of memory usage measured in Gigabytes (GB). The first two metrics define the volume of traffic arriving from each delivery route and help us identify the data reduction due to edge processing being executed at the E2GW. Moreover, they help us identify the usage frequency of the DDF for detecting duplicate deliveries to support *"at-most-once"* and *"exactly-once"* QoS guarantees.

4.2.3. First scenario: baseline performance of legacy elements

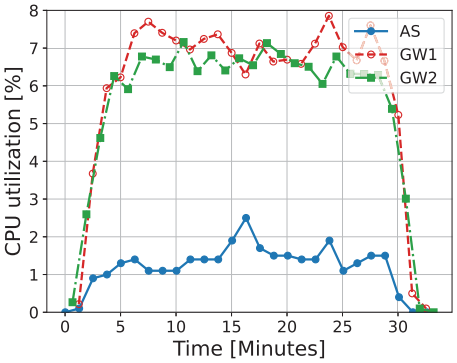
We start by measuring the performance of the LoRaWAN deployment when only the legacy components are activated. Fig. 11 depicts the overall performance in terms of the performance metrics defined in the previous section. Since the devices are progressively activated, during the first five minutes the number of frames received by each gateway and consequently transmitted to the AS increases. During the next period of 20 min, given that all the devices are activated, we observe a steady reception of frames from both gateways and their forwarding to the AS. During this period, all devices collectively transmit 1000 frames per second. Finally, during the last 5 min of the experiments, after having transmitted all 500 frames, the devices disactivate. We thus see that during this last period the total number of frames received by the gateways eventually drops to zero.

Looking into the period where all the devices are activated in Fig. 11(a), we note that although a total of 1000 frames are transmitted every second by all the devices, we observe that each gateway correctly receives approximately 300 frames per second. The number of lost frames is due to the integrated FDR model used in this study, as presented in Section 3, which is about 31% at each GW, equivalent to approximately 300 frames per second. This is an indicator of the number of transmission failures at the wireless medium due to interference resulting from the given device density. This kind of visualization is essential for understanding the load distribution among different gateways in the network.

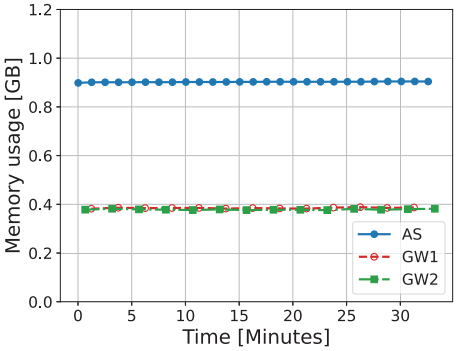
Among these 300 frame transmissions properly received by each gateway every second, some frames are received by only one gateway while others are received by both. Those frames received by both gateways are detected as duplicate frames when they



(a) Frame rate received per second by each E2GW and by the AS



(b) CPU usage of the host machines accommodating the E2GW and the AS



(c) Memory usage of the host machines accommodating the E2GW and the AS

Fig. 12. Overall performance for E2ED, E2GW and AS when Edge2LoRa extensions are activated via the TheThingsNetwork public infrastructure.

reach the NS and are thus dropped. Therefore these frames represent the union probability of the frames received at each GW and expressed as $P(GW_1)$ and $P(GW_2)$ respectively, and computed $P(GW_1 \cup GW_2) = P(GW_1) + P(GW_2) - (P(GW_1) \cap P(GW_2))$, where we take in to account the intersect probability representing the duplicate frames. This equation accounts for the probability of frames received at both gateways, ensuring no double counting of the overlapping frames removed by the NS. Finally, the AS receives approximately 520 frames per second, that accounts for about the 50% of the frames emitted from the emulated devices.

During the experiment, we observed that the resource utilization of both GWs and also for the AS are maintained at minimum levels. In terms of computational load, Fig. 11(b) the CPU usage is lower in the host machine accommodating the AS, average of 1.8% concerning 3.2%, due to the more powerful hardware. In Fig. 11(c) we observe the memory usage at the host machines that accommodate the GW and the AS. Again, the memory usage at the AS is higher than at the GWs. This is because it runs an Ubuntu operating system, as opposed to the Raspbian operating system deployed on the two host machines where the GWs are deployed. The Ubuntu system is more resource-intensive, which contributes to the observed higher memory usage of 1 GB with respect to the 0.4 GB perceived from the host machines accommodated by the GWs.

4.2.4. *Second scenario: performance of the Edge2LoRa extensions*

We now proceed with the performance evaluation when the Edge2LoRa extensions are activated. Fig. 12(a) depicts the number of frames received by each E2GW, which is more or less the same as in the case when legacy GWs were used as depicted in Fig. 11(a). On the other hand, the number of frames transmitted to the AS is drastically reduced due to the edge-computing tasks deployed to the E2GW module executor. In comparison to the legacy scenario, the number of frames received by the AS is reduced to approximately 10%: on average, about 52 messages compared to 520 frames. This drastic reduction illustrates the efficiency of the aggregation function, which can help conserve network resources and reduce data transmission volume without losing essential information.

In terms of the resulting computational load of the E2GW when the Edge2LoRa components are activated, Fig. 12(b) indicates an increase on the CPU usage by about an average of 6.8 percentage points. This increase in the CPU usage is due to the edge-computing

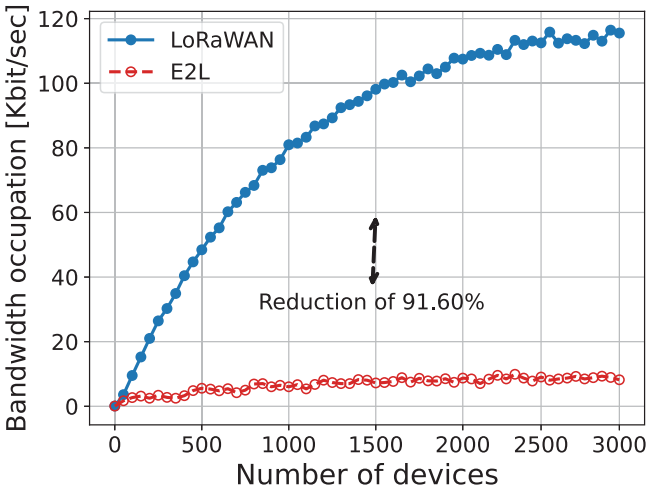


Fig. 13. Assessment of the benefits for the proposed approach in terms of bandwidth utilization using the legacy LoRaWAN standard as baseline in comparison with the Edge2LoRa extensions when the edge-computing task is deployed to the E2GW.

task being executed that carries out a series of one-to-one and many-to-one operators on the 30-seconds windows. Moreover, the increase in the CPU usage is also due to the deciphering and ciphering of each frame so that the data payload can be extracted. Given all these considerations, the shift of processing from the cloud to the edge does not result in any significantly additional resource consumption in terms of CPU usage. On the other hand, the CPU usage of the AS is slightly reduced by an average of 1.2 percentage points since the AS is no longer aggregating the frames as this is now done at the edge of the network. Similarly, in terms of memory usage, it seems that the activation of the Edge2LoRa components does not present any significant increase when compared with the legacy scenario.

We believe that these experiments provide evidence that the Edge2LoRa extensions can help reduce the network traffic of the core backbone network and also reduce the processing load on the AS, without significantly increasing resource utilization at the edge level. This finding is crucial as it demonstrates the potential of edge computing when combined with LoRaWAN in managing network resources more efficiently.

To further illustrate the benefits provided by the Edge2LoRa approach, we look into the first five minutes of the experiment as the number of devices are being activated. We calculate the bandwidth utilization of the IP-level network backbone connecting the gateways with the NS of the public infrastructure. Fig. 13 depicts the drastic reduction in bandwidth utilization as the number of active devices increases from 1 to 3000. Using this specific reference, we can deduce that there is an average bandwidth reduction of 91.60% when the number of active E2ED is 1500. This demonstrates that the Edge2LoRa approach, with its strong data aggregation capabilities at the edge level, can significantly reduce network bandwidth usage. This is particularly beneficial in large-scale deployments or high-density environments with numerous active devices, where the gateways utilize internet connections provided by various access technologies, e.g. 3G/4G access technology with lower bandwidth. This also applies to situations where multiple gateways share the same backhaul through a single central NS.

4.2.5. Third scenario: verifying the backwards compatibility of the Edge2LoRa

The previous experiments were conducted by utilizing a public infrastructure, that is TheThingsNetwork. We wish to look further into the backwards compatibility of the Edge2LoRa extensions by conducting an experiment in a scenario with 1500 legacy EDs and 1500 E2ED deployed over the same area where two E2GW are operating. In this case, the Edge2LoRa driver equally assigns the 1500 E2ED to the two E2GW. At the same time, the 1500 legacy EDs are connected by both E2ED. Fig. 14 depicts the number of frames received by each E2GW. The results are substantially similar to before for what concerns the received frames at each E2GW. However, the number of frames received at the AS falls somewhere in between the values observed in purely legacy and Edge2LoRa scenarios. Since only half of the deployed devices are E2ED, the edge-computing task executed at the two E2GW can only aggregate the sensor values included in the frames transmitted by the 1500 E2ED. Specifically, since the E2ED transmit on average 500 frames per second, these are reduced, on average, to 50 messages that are transmitted to the AS. The remaining 500 frames per second received on average by the legacy EDs, are directly forwarded for processing to the AS through the NS. We remark here that a subset of these frames are not received due to the interference phenomena. This scenario illustrates how the system can handle both legacy EDs and E2ED devices simultaneously, eliminating the need to replace existing devices during a transition period. This feature enhances the system's flexibility and adaptability, making it more feasible for real-world implementations.

We also wish to look into the latency of data reaching the AS. As shown in Fig. 2, the direct connection between the gateway layer and the AS avoids the need for data to cross the NS, thereby reducing total latency since the NS is not involved in the processing of the frames. To provide some quantitative evidence of this reduction on the processing overhead of the packets, we use private infrastructure where NS, AS and E2GW are within the same local network thus minimizing the delays due to the transmission of the

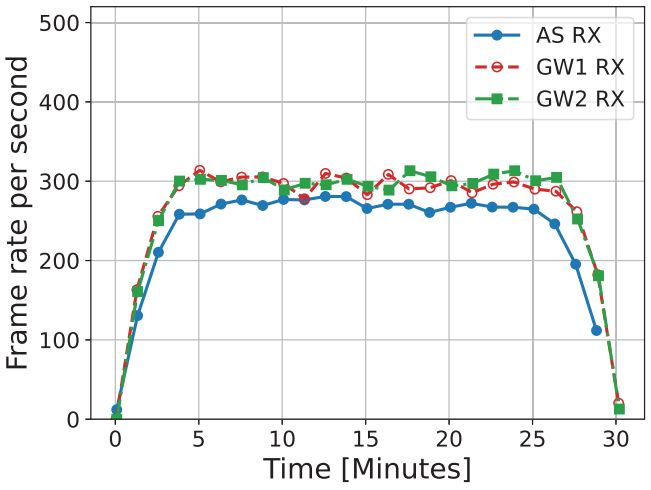


Fig. 14. Assessment of the backwards compatibility of the proposed approach.

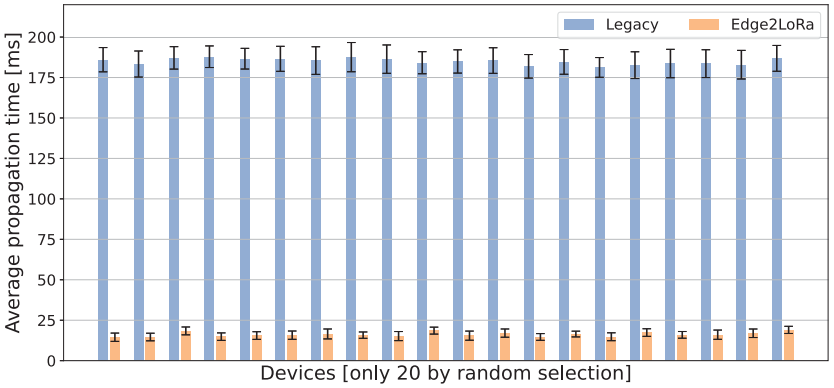


Fig. 15. Impact of the proposed method on data latency to the AS. The bar plot represents the average latency for each terminal in the legacy scenario, with the standard deviation indicated by the error bar.

frames over the network. We randomly select 20 out of the 3000 terminals and plot the average delay time comparison between the legacy and Edge2LoRa scenarios. The resulting figures are depicted in Fig. 15. In the legacy case, we observe that the average time for the frames that arrive at the gateway to final reach the AS while passing through the NS is approximately 180 ms, depicted as a bar plot where the standard deviation is also presented via the error bar. On the other hand, for the case of Edge2LoRa, due to the direct connection of the E2GW with the AS for the same 20 devices configured as E2ED, the average time drops to approximately 16 ms. Remark that in the legacy case, the aggregation of the frames is carried out at the AS while for the Edge2LoRa case the aggregation takes place in the E2GW. By comparing the two scenarios, we observe a reduction of 92% in delay when data streams are moved directly from the gateways to the AS.

4.2.6. Fourth scenario: handover in case of failures

We now wish to look into the case when the E2GW experience failures. During the execution of the experiment where all the devices are E2ED and both gateways are E2GW, we introduce a failure occurring at one of the two E2GW at the 15th minute. We expect that at some point the Edge2LoRa driver will notice that the gateway registry has marked the failing E2GW as being offline and re-assign the E2ED to the other E2GW. Fig. 16 shows how the frame rate of the failing E2GW drops to zero while the traffic of the other E2GW almost doubles after the re-assignment of the E2EDs. The experiment illustrates the resilience of the proposed approach. It is evident that the processed frames at E2GW 1 drop to zero immediately at the 15-minute mark. Despite this, the number of received frames at the sink perceived a lower influence because all the E2ED are now managed from the E2GW 2.

4.3. Security analysis

We now proceed with the analysis of the security properties of the Edge2LoRa extensions. We focus on the three cryptographic properties that the group key establishment protocol shall guarantee, which are: (i) the computational key secrecy; (ii) the decisional

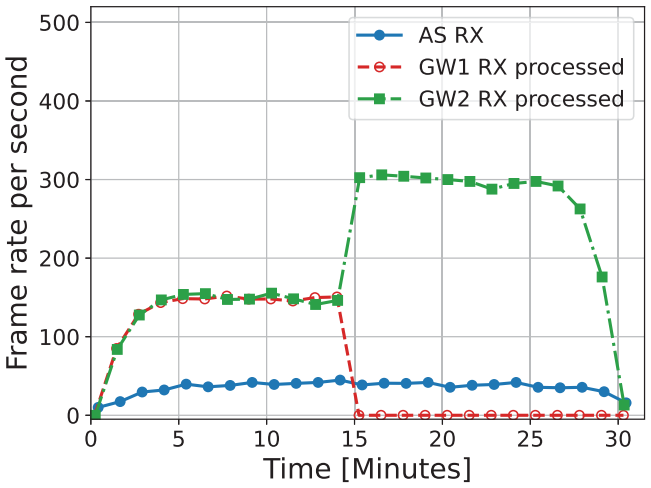


Fig. 16. Simulating a failure of E2GW 1 after 15 min of activity to showcase the resilience of our proposed method.

key secrecy; (iii) key independence. The Edge2LoRa security analysis including the three properties is reported in the following sections.

4.3.1. Computational & decisional key secrecy

It must be computationally infeasible for any passive adversary to discover any key, while no information may be leaked other than the public key. The security of the protocol is based on the use of ECDH, an algorithm that enables secure communication between two parties over an insecure channel. The security of ECDH is derived from the computational difficulty of solving the Elliptic Curve Discrete Logarithm problem. This problem involves finding the discrete logarithm of a point on an elliptic curve, which is considered computationally difficult to solve. The large size of the elliptic curve group and the complex mathematical operations involved make it challenging for an attacker to derive the private key from the public key [43]. By utilizing ECDH, the Edge2LoRa ensures the confidentiality and integrity of the communication, making it resistant to attacks attempting to compromise the shared secret key.

To evaluate the computational and decisional key secrecy, we consider a Man-in-the-Middle (MITM) attack scenario. In the proposed protocol, an attacker attempting a MITM attack would be unable to eavesdrop on the LoRaWAN frames because they are encrypted using the *AppSEncKey*. Even if an attacker manages to break this key by exploiting a vulnerability in the OTAA process, it would still be computationally challenging to guess the *EdgeSKey*.

4.3.2. Key independence

A passive adversary that manages to acquire a subset of the keys must not be able to discover any other information about the remaining keys. This property is further decomposed into:

- Forward Secrecy: A passive adversary that knows a subset of keys must not discover any subsequent keys.
- Backward secrecy: A passive adversary that knows a subset of keys must not discover any preceding keys.

To enhance the overall security of the activation methods, our proposal offers a way to periodically refresh the **Edge Session Keys** by performing a new execution of the protocol. Since a new pair of ephemeral ECC keys is computed by every actor in each execution of the protocol, the key independence is guaranteed. There is no correlation between the previous set of keys and the new one, or any subsequent ones.

Remark that only a single uplink and downlink LoRaWAN frame are needed for the re-execution of the protocol, it does not significantly impact the performance and energy consumption of the E2ED.

Finally, it is important to note that no security analysis of the communication over IP-based networks is performed, as our solution does not impose any specific protocol requirements.

5. Related work

LoRaWAN has received significant attention during the past years since it is a crucial and promising LPWAN technology [16]. A comprehensive review of LoRaWAN from the perspective of network operation, security, applications, modeling and experimentation the interested reader is presented in [10]. On the other hand, during the past couple of years, to the best of our knowledge, only a handful of studies have been proposed to address the centralized architecture of the LoRaWAN network standard. In the rest of this section we will try to summarize all the relevant studies.

The design of a LoRaWAN deployment in a smart campus enabling fog computing on the LoRaWAN gateways giving them the LoRaWAN servers capabilities is presented in [17]. The evaluation shows the benefits of the use of fog computing however the solution proposed is not compatible with existing public networks and its scalability is limited since the functionalities of the LoRaWAN GWs and NS are collapsed in a single entity. The focus of the authors was limited to the use of fog computing nodes to support low-latency and location-aware IoT applications, and the evaluation was conducted on a particular smart campus. As a result, there are no indications provided on the scalability of the solution.

An alternative approach is proposed in [18] that decouples the functionalities of the NS into four modules: connector, central server, join server and network controller. Such a split can indeed assist in distributing the centralized functionalities to different locations within the network. The scalability however of this approach remains limited, the functionalities that can be decentralized are only related to the network management and operation, while at the same time, the proposed extensions are not compatible with existing, public LoRaWAN deployments. A similar approach is followed in [19] where it is proposed to migrate the functions of rejoin and media access control (MAC) commands into the GW however instead of using a local database, a blockchain network with multiple ledgers is designed. The resulting system improves the overall scalability of the network operations by utilizing the edge resources available at the LoRa GW, however, yet again it is not backward compatible with the current LoRaWAN standard. Contrary to these studies, our solution enables Edge/Fog Computing over LoRaWAN at both the radio and application levels. We are therefore able to exploit the same benefits mentioned earlier in terms of network management, while also extending these advantages to data-level operations.

The LoRaCTP is a flexible protocol based on LoRa that enables data transfer over large distances with very low energy expenditure over a generic FrUgal eDGE (FUDGE)/fog architecture [20]. The solution provides all necessary mechanisms for LoRaWAN reliability and allows edge computing via a lightweight connection, it is not however compatible with existing LoRaWAN deployments.

The ability to introduce edge/fog processing for the analysis of the payload of LoRaWAN frames is studied in [38,39]. The proposed architectures utilize the computational resources already available at the LoRa GW deployed to facilitate the analysis of IoT data in smart water distribution networks. A distributed key management system is introduced that shares the Application Session Encryption Key (*AppSEncKey*) with the LoRaWAN GWs so that they can decrypt and analyze the application-level messages received. Both studies indicate significant reductions to the overall load to network and cloud resources. However, transmitting the *AppSEncKey* to the disparate LoRa GWs introduces new vulnerabilities to the system. In contrast to these approaches our solution does not require the exchange of any secret key across the network, Edge2LoRa offers a group key agreement that allows a high security standard allowing confidentiality and non-repudiation of both radio-level and application-level data and metadata.

A different approach is followed in [44] to minimize upstream network traffic in cases of overlapping network areas covered by multiple nearby GWs. In LoRaWAN frames received by one or more GW are forwarded multiple times through the network backbone to a central NS. To reduce the duplicate transmission, each ED selects the nearby GW with which it has the best signal quality to act as a so-called Rendezvous Point where analysis of the frames payload can also take place. The authors evaluate simulating dense deployments, in OMNet++, enlightening certain limitations since transmitting a large number of downlink frames leads to high loss rates. In contrast to this approach, we propose a scalable E2GW selection that does not require the exchange of any additional downlink frames, and our solution tests large network deployments using a real implementation of Edge2LoRa.

6. Conclusions

The ever-growing resource needs of modern-day large-scale IoT deployments regarding guaranteed low latency and the massive data transfer rate are constantly pushing the boundaries of technologies and requiring a paradigm shift from the traditional producer/consumer model. At the same time, the new paradigm called cloud edge computing continuum (CECC) that evolves beyond the more traditional central cloud/DC with ultra-high-end processing powers and high-capacity networking infrastructure to extend their coverage all the way to the network edge, has not yet been fully extended to include edge and far-edge resources available in IoT deployments. This is mainly because LoRaWAN adopts the design approach of using simple protocols to realize a centralized architecture that guarantees the security and confidentiality of the IoT generated data. Forcing the gateways to act as simple bridges, at one hand accommodates the rapid and low-cost deployment of unlicensed LPWANs, on the other hand excludes them from potentially acting as trusted intermediate processing and storage elements in the CECC.

Recently some attempts have been made in modifying the specified operation of the LoRaWAN standard protocols, proposing alternative architectures to reduce the significant pressure imposed on the central cloud services in cases of massive IoT data streams or time-constrained consumption of IoT data. At the same time, introducing changes to the architecture and the specified operation of the protocols while maintaining backward compatibility is a challenging discourse. As a result, to the best of our knowledge, there is currently no proposal that incorporates LoRaWAN in the CECC both for network management and application data processing that maintains the practical philosophy of LoRaWAN and respects backward compatibility.

In this paper, we described Edge2LoRa, an enhanced version of LoRaWAN that transforms LoRa gateways from simple bridges to edge processing units in a secure and privacy preserving manner. By taking the advantage of edge/fog computing, Edge2LoRa offers substantial benefits to allow the execution of Big Data analytics across the CECC over LoRaWAN. The proposed design enables dynamic cloud–edge–far-edge coordination based on network conditions and application-level parameters. The resulting system ensures backward compatibility for seamless interoperability between legacy and new elements without disrupting network operation and providing quality of service guarantees.

We present a comprehensive design of the key components and rational choices within the Edge2LoRa. Our description points several key features of Edge2LoRa and how it differs from standard LoRaWAN. We provide specific implementation decisions that

result into an efficient, secure, and adaptable infrastructure for optimizing sensor data stream processing in LoRaWAN networks. Edge2LoRa does not limit the network operator to this particular usage. Other design choices are also possible so that operation is optimized based on different network topologies, device characteristics and Big Data analytics.

Secure communication is ensured through shared session encryption keys within the Edge2LoRa system, with Elliptic Curve Cryptography proposed for creating these cryptographic keys. ECC offers high security with lower resource consumption, making it suitable for resource-constrained environments like LoRaWAN EDs with minimal energy consumption.

A performance evaluation was conducted to assess the correctness, security, and network utilization of the new system. This evaluation approach considers the co-existence of legacy EDs and GWs with new ones, creating the possibility of unprocessed frames arriving at the NS/AS via legacy equipment, while simultaneously, some parts of the data arrive at the AS after being processed by an Edge2LoRa GW.

Adopting Edge2LoRa could lay the groundwork for more efficient, scalable, and secure LPWAN deployments, enabling IoT networks to effectively manage the increasing demands of real-world applications. In future works, we aim to enhance the scalability of Edge2LoRa by supporting both horizontal and vertical scaling through the decomposition of the functionalities offered by the E2GW into various micro-services. This approach will enable multiple applications to operate concurrently on the same edge layer (vertical scaling) and simplify the process of deploying services across multiple gateways (horizontal scaling). This will facilitate the containerization of different services, making deployment and management more flexible and efficient. We would also like to evaluate the total network stability in more complex deployments with more than two gateways.

CRediT authorship contribution statement

Stefano Milani: Writing – original draft, Validation, Investigation, Formal analysis, Data curation, Conceptualization. **Domenico Garlisi:** Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Carlo Carugno:** Visualization, Software, Data curation. **Christian Tedesco:** Writing – original draft, Visualization, Software, Data curation. **Ioannis Chatzigiannakis:** Writing – review & editing, Writing – original draft, Supervision, Resources, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Ioannis Chatzigiannakis and Domenico Garlisi report financial support was provided by Horizon Europe and the NRRP MUR program funded by the EU-NGEU. The other authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgments

This work was partially supported by the project SERICS (PE00000014 - CUP D33C22001300002) under the NRRP MUR program funded by the EU-NGEU. This work was partially supported by the project RESTART (PE00000001 - CUP E83C22004640001) under the NRRP MUR program funded by the EU-NGEU. This paper was partially supported by the European Union's Horizon 2020 research and innovation programme under grant Agreement 957286, ELEGANT project.

References

[1] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, W. Zhao, A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications, IEEE Internet Things J. 4 (5) (2017) 1125–1142, http://dx.doi.org/10.1109/JIOT.2017.2683200, URL https://ieeexplore.ieee.org/document/7879243.

[2] M. Zorzi, A. Gluhak, S. Lange, A. Bassi, From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view, IEEE Wirel. Commun. 17 (6) (2010) 44–51, http://dx.doi.org/10.1109/MWC.2010.5675777, URL https://ieeexplore.ieee.org/document/5675777.

[3] M. Jouhari, N. Saeed, M.-S. Alouini, E.M. Amhoud, A survey on scalable LoRaWAN for massive IoT: Recent advances, potentials, and challenges, IEEE Commun. Surv. Tutor. 25 (3) (2023) 1841–1876, http://dx.doi.org/10.1109/COMST.2023.3274934.

[4] X. Chen, D.W.K. Ng, W. Yu, E.G. Larsson, N. Al-Dhahir, R. Schober, Massive access for 5G and beyond, IEEE J. Selected Areas Commun. 39 (3) (2021-03) 615–637, http://dx.doi.org/10.1109/JSAC.2020.3019724, URL https://ieeexplore.ieee.org/document/9205230.

[5] U. Raza, P. Kulkarni, M. Sooriyabandara, Low Power Wide Area networks: An overview, IEEE Commun. Surv. Tutor. 19 (2) (2017) 855–873, http://dx.doi.org/10.1109/COMST.2017.2652320.

[6] D. Garlisi, A. Pagano, F. Giuliano, D. Croce, I. Tinnirello, A coexistence study of low-power wide-area networks based on LoRaWAN and sigfox, in: 2023 IEEE Wireless Communications and Networking Conference (WCNC), 2023-03, pp. 1–7, http://dx.doi.org/10.1109/WCNC55385.2023.10118692, URL https://ieeexplore.ieee.org/document/10118692.

[7] J. Haxhibeqiri, E. De Poorter, I. Moerman, J. Hoebeke, A survey of LoRaWAN for IoT: From technology to application, Sensors 18 (11) (2018) http://dx.doi.org/10.3390/s18113995.

[8] LoRa Alliance, LoRaWAN 1.1 specification, 2017, technical specification.

[9] A. Pagano, D. Croce, I. Tinnirello, G. Vitale, A survey on LoRa for smart agriculture: Current trends and future perspectives, IEEE Internet Things J. 10 (4) (2023) 3664–3679, http://dx.doi.org/10.1109/JIOT.2022.3230505.

[10] Z. Sun, H. Yang, K. Liu, Z. Yin, Z. Li, W. Xu, Recent advances in LoRa: A comprehensive survey, ACM Trans. Sen. Netw. 18 (4) (2022) http://dx.doi.org/10.1145/3543856.

[11] ETSI, Final draft ETSI EN 300 220-1 V2.4.1 (2012-01), Technical Report REN/ERM-TG28-434, 2012, https://www.etsi.org/deliver/etsi_tr/103500_103599/103526/01.01.01_60/tr_103526v010101p.pdf.

[12] T. Janssen, N. BniLam, M. Aernouts, R. Berkvens, M. Weyn, LoRa 2.4 GHz communication link and range, Sensors 20 (16) (2020) http://dx.doi.org/10.3390/s20164366, URL https://www.mdpi.com/1424-8220/20/16/4366.

[13] F. Adelantado, X. Vilajosana, P. Tuset-Peiro, B. Martinez, J. Melia-Segui, T. Watteyne, Understanding the limits of LoRaWAN, IEEE Commun. Mag. 55 (2017) 34–40.

[14] W. Rafique, L. Qi, I. Yaqoob, M. Imran, R.U. Rasool, W. Dou, Complementing IoT services through software defined networking and edge computing: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 1761–1804, http://dx.doi.org/10.1109/COMST.2020.2997475, URL https://ieeexplore.ieee.org/document/9099866.

[15] C. Mouradian, D. Naboulsi, S. Yangui, R.H. Glitho, M.J. Morrow, P.A. Polakos, A comprehensive survey on fog computing: State-of-the-art and research challenges, IEEE Commun. Surv. Tutor. 20 (2018) 416–464, http://dx.doi.org/10.1109/COMST.2017.2771153.

[16] V.K. Sarker, J.P. Queralta, T.N. Gia, H. Tenhunen, T. Westerlund, A survey on LoRa for IoT: Integrating edge computing, in: 2019 Fourth International Conference on Fog and Mobile Edge Computing, FMEC, 2019, pp. 295–300, http://dx.doi.org/10.1109/FMEC.2019.8795313, URL https://ieeexplore.ieee.org/document/8795313.

[17] P. Fraga-Lamas, M. Celaya-Echarri, P. Lopez-Iturri, L. Castedo, L. Azpilicueta, E. Aguirre, M. Suárez-Albela, F. Falcone, T.M. Fernández-Caramés, Design and experimental validation of a LoRaWAN fog computing based architecture for IoT enabled smart campus applications, Sensors 19 (15) (2019) 3287, http://dx.doi.org/10.3390/s19153287, URL https://www.mdpi.com/1424-8220/19/15/3287.

[18] Q. Zhou, K. Zheng, L. Hou, J. Xing, R. Xu, Design and implementation of open LoRa for IoT, IEEE Access 7 (2019) 100649–100657, http://dx.doi.org/10.1109/ACCESS.2019.2930243.

[19] L. Hou, K. Zheng, Z. Liu, X. Xu, T. Wu, Design and prototype implementation of a blockchain-enabled LoRa system with edge computing, IEEE Internet Things J. 8 (4) (2021) 2419–2430, http://dx.doi.org/10.1109/JIOT.2020.3027713.

[20] K. Nakamura, P. Manzoni, A. Redondi, E. Longo, M. Zennaro, J.-C. Cano, C.T. Calafate, A LoRa-based protocol for connecting IoT edge computing nodes to provide small-data-based services, Digit. Commun. Netw. 8 (3) (2022) 257–266, http://dx.doi.org/10.1016/j.dcan.2021.08.007, URL https://linkinghub.elsevier.com/retrieve/pii/S2352864821000596.

[21] D. Garlisi, I. Tinnirello, G. Bianchi, F. Cuomo, Capture aware sequential waterfilling for LoRaWAN adaptive data rate, IEEE Trans. Wireless Commun. 20 (3) (2021) 2019–2033, http://dx.doi.org/10.1109/TWC.2020.3038638.

[22] S. Zeuch, A. Chaudhary, B. Monte, H. Gavriilidis, D. Giouroukis, P. Grulich, S. Breß, J. Traub, V. Markl, The NebulaStream platform: Data and application management for the internet of things, in: Conference on Innovative Data Systems Research, CIDR, 2020.

[23] M. Xekalaki, J. Fumero, A. Stratikopoulos, K. Doka, C. Katsakioris, C. Bitsakos, N. Koziris, C. Kotselidis, Enabling transparent acceleration of big data frameworks using heterogeneous hardware, Proc. VLDB Endow. 15 (13) (2022) 3869–3882, http://dx.doi.org/10.14778/3565838.3565842.

[24] I. Butun, N. Pereira, M. Gidlund, Analysis of LoRaWAN v1.1 security, in: Proceedings of the 4th ACM MobiHoc Workshop on Experiences with the Design and Implementation of Smart Objects - SMARTOBJECTS '18, ACM Press, New York, New York, USA, 2018, pp. 1–6, http://dx.doi.org/10.1145/3213299.3213304.

[25] S. Milani, I. Chatzigiannakis, Design, analysis, and experimental evaluation of a new secure rejoin mechanism for LoRaWAN using elliptic-curve cryptography, J. Sens. Actuator Netw. 10 (2021) 36, http://dx.doi.org/10.3390/jsan10020036.

[26] E. Aras, G.S. Ramachandran, P. Lawrence, D. Hughes, Exploring the security vulnerabilities of LoRa, in: 2017 3rd IEEE International Conference on Cybernetics (CYBCONF), IEEE, 2017, pp. 1–6, http://dx.doi.org/10.1109/CYBConf.2017.7985777.

[27] K. Lauter, The advantages of elliptic curve cryptography for wireless security, IEEE Wirel. Commun. 11 (2004) 62–67, http://dx.doi.org/10.1109/MWC.2004.1269719.

[28] D. McGrew, K. Igoe, M. Salter, Fundamental elliptic curve cryptography algorithms, Internet Engineering Task Force RFC 6090, 2011, pp. 1–34.

[29] R.R. Ahirwal, M. Ahke, Elliptic curve diffie-hellman key exchange algorithm for securing hypertext information on wide area network, Int. J. Comput. Sci. Inf. Technol. 4 (2013) 363–368.

[30] R. Géraud-Stewart, M. Lombard-Platet, D. Naccache, Approaching optimal duplicate detection in a sliding window, in: Computing and Combinatorics: 26th International Conference, COCOON 2020, Atlanta, GA, USA, August 29–31, 2020, Proceedings 26, Springer, 2020, pp. 64–84.

[31] S. Milani, D. Garlisi, M. Di Fraia, P. Pisani, I. Chatzigiannakis, Enabling edge processing on LoRaWAN architecture, in: Proceedings of the 29th Annual International Conference on Mobile Computing and Networking, ACM, Madrid Spain, 2023, pp. 1–3, http://dx.doi.org/10.1145/3570361.3614074, URL https://dl.acm.org/doi/10.1145/3570361.3614074.

[32] RIOT-OS, RIOT-OS, 2020, https://github.com/RIOT-OS/RIOT.

[33] M. Rottleuthner, T.C. Schmidt, M. Wählisch, Eco: A hardware-software co-design for in situ power measurement on low-end IoT systems, in: Proceedings of the 7th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems, 2019, pp. 22–28.

[34] kmackay, Micro-ecc, 2020, https://github.com/kmackay/micro-ecc.

[35] RIOT-OS, RIOT OS Crypto module, https://api.riot-os.org/group_sys_crypto.html.

[36] C. Gündoğan, C. Amsüss, T.C. Schmidt, M. Wählisch, IoT content object security with OSCORE and NDN: a first experimental comparison, in: 2020 IFIP Networking Conference (Networking), IEEE, 2020, pp. 19–27.

[37] T. Kolajo, O. Daramola, A. Adebiyi, Big data stream analysis: a systematic literature review, J. Big Data 6 (1) (2019) 47.

[38] D. Amaxilatis, I. Chatzigiannakis, C. Tselios, N. Tsironis, N. Niakas, S. Papadogeorgos, A smart water metering deployment based on the fog computing paradigm, Appl. Sci. 10 (6) (2020) http://dx.doi.org/10.3390/app10061965.

[39] D. Garlisi, G. Restuccia, I. Tinnirello, F. Cuomo, I. Chatzigiannakis, Leakage detection via edge processing in lorawan-based smart water distribution networks, in: 2022 18th International Conference on Mobility, Sensing and Networking, MSN, 2022, pp. 223–230, http://dx.doi.org/10.1109/MSN57253.2022.00047.

[40] O. Akrivopoulos, N. Zhu, D. Amaxilatis, C. Tselios, A. Anagnostopoulos, I. Chatzigiannakis, A fog computing-oriented, highly scalable iot framework for monitoring public educational buildings, in: 2018 IEEE International Conference on Communications, ICC, IEEE, 2018, pp. 1–6.

[41] System reference document (srdoc); technical characteristics for low Power Wide Area networks chirp spread spectrum (LPWAN-CSS) operating in the UHF spectrum below 1 GHz, Tech. Rep. 103 526, ETSI, 2018, v1.1.1.

[42] L. Bhatia, P.-Y. Chen, M. Breza, C. Zhao, J.A. McCann, IRONWAN: Increasing reliability of overlapping networks in LoRaWAN, IEEE Internet Things J. (2021).

[43] B. Shani, On the bit security of elliptic curve diffie-hellman, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 10174 LNCS, Springer Verlag, 2017, pp. 361–387, http://dx.doi.org/10.1007/978-3-662-54365-8_15/COVER.

[44] I. Fardin, S. Milani, F. Cuomo, I. Chatzigiannakis, Enabling edge computing over LoRaWAN: A device-gateway coordination protocol, in: Proceedings of the 12th ACM International Symposium on Design and Analysis of Intelligent Vehicular Networks and Applications, DIVANet '22, 2022, pp. 23–30, http://dx.doi.org/10.1145/3551662.3560926.