electronics

Article

A Modular IoT Hardware Platform for Distributed and Secured Extreme Edge Computing

Pablo Merino, Gabriel Mujica *, Jaime Señor and Jorge Portilla

Centro de Electrónica Industrial, Universidad Politécnica de Madrid, José Gutiérrez Abascal 2, 28006 Madrid, Spain; p.merino@upm.es (P.M.); jaime.senors@alumnos.upm.es (J.S.); jorge.portilla@upm.es (J.P.)

* Correspondence: gabriel.mujica@upm.es; Tel.: +34-910-676-944

Received: 25 February 2020; Accepted: 18 March 2020; Published: 24 March 2020

Abstract: The hardware of networked embedded sensor nodes is in continuous evolution, from those 8-bit MCUs-based platforms such as Mica, up to powerful Edge nodes that even include custom hardware devices, such as FPGAs in the Cookies platform. This evolution process comes up with issues related to the deployment of the Internet of Things, particularly in terms of performance and communication bottlenecks. Moreover, the associated integration process from the Edge up to the Cloud layer opens new security concerns that are key to assure the end-to-end trustability and interoperability. This work tackles these questions by proposing a novel embedded Edge platform based on an EFR32 SoC from Silicon Labs with Contiki-NG OS that includes an ARM Cortex M4 MCU and an IEEE 802.15.4 transceiver, used for resource-constrained low-power communication capabilities. This IoT Edge node integrates security by hardware, adding support for confidentiality, integrity and availability, making this Edge node ultra-secure for most of the common attacks in wireless sensor networks. Part of this security relies on an energy-efficient hardware accelerator that handles identity authentication, session key creation and management. Furthermore, the modular hardware platform aims at providing reliability and robustness in low-power distributed sensing application contexts on what is called the Extreme Edge, and for that purpose a lightweight multi-hop routing strategy for supporting dynamic discovery and interaction among participant devices is fully presented. This embedded algorithm has served as the baseline end-to-end communication capability to validate the IoT hardware platform through intensive experimental tests in a real deployment scenario.

Keywords: extreme edge; embedded edge computing; internet of things deployment; hardware design; IoT security; Contiki-NG; trustability

1. Introduction

The Internet of Things paradigm has achieved an enormous integration level inside the technology distributed all around the world. It covers from consumer electronics, such as Wi-Fi controlled thermostats, to industrial or professional applications, such as a Wireless Sensor Network (WSN) registering data all along a whole forest. The future of communication protocols could also lead to bigger growths on new IoT system implementations, e.g., real-time systems collecting data from dozens of sensors around a fabric to optimize manufacturing operations dynamically. Therefore, the development of this kind of platforms looks promising.

The traditional hardware solutions that were used in Wireless Sensor Networks mostly relied on de-facto standard sensor motes, such as Micas and TelosB [1], or similar approaches where 8-bit or 16-bit-based microcontrollers were integrated as the core of the wireless devices, to perform simple yet energy-efficient tasks for the target application. During the last decade tens of hardware platforms for the Extreme Edge [2] of the IoT have appeared with different elements and focusing on aspects such as low power consumption, high processing capabilities or open HW philosophy, among others [3]. Although these hardware platforms have been valid for many WSN application contexts, the ongoing revolution of IoT is pushing the hardware implementation towards the integration of more complex capabilities that allow tackling the challenges of smart and highly dynamic scenarios [4], particularly concerning the arising end-to-end IoT security issues with such an amount of expected Edge devices in place. In this sense, the traditional behavior of a WSN, in which the devices remained in sleep modes for a very long period of time (thus the main consumption components to be considered were the deep power modes) and then wake up to transmit a sensing measurement to a root device, is changing to more active collaborations among participant sensors, in which the type of features they provide to the local or overall IoT Edge deployment becomes a key performance element of the system. In such scenarios, protecting the relationships among the nodes is crucial to assure data integrity and security on the Edge.

Nonetheless, it seems that practical implementation problems of IoT networks are always related to security and reliability issues. One of the many reasons is that those networks are usually built up from many nodes that have some restrictions on energy consumption and processing capabilities. Thus, they are often designed as tiny embedded systems with low-cost processors that do not have the enough computational power to implement security systems. However, if the problems associated with a lack of security capabilities are ignored, several disasters on the network and on the products related to it can indeed appeared. A common mistake on simple IoT products deployment is to think that data exchanged by the nodes on the Edge is not critical, as it does not contain private data about users, e.g., humidity sensors sensing moisture measurements to the central node that controls the overall climate of one house. However, since the security process is too simple or may not even exist on these nodes, an attacker could take control of one node and then scale privileges through upper layers of the network, reaching cloud server and the data located there.

The problem resides not only in the ability of the attacker to scale privileges in the network. Actions taken inside the nodes on the Edge might be also harmful for the system and for the users. Although many people might think that an attacker stealing data related to the temperature of their rooms is not a real threat, other systems may suffer from this uncontrolled access to the Edge. A good example of that is the research made by the authors in [5], where they analyzed the security problems within the Philips Hue lamps. In this case, the authors were able to infer the key that protects the firmware updates of the lamps, by measuring the patterns on the power consumption of the main chip while making cryptographic tasks. Then, they could upload new custom versions of the firmware to the lamps by just requesting it to the chip, once a minimum distance from the node is reached. Upon this update being accepted, the new firmware can be programmed to request the same update to other lamps on the network, propagating itself such as an infection. The ability to change the software that controls the nodes is the key to allow the attackers to cause several problems to the users. Looking at those lamps, the authors remark the possibility of causing epileptic attacks to users by generating stroboscopic lights. In this context, some solutions are being proposed by the community [6] in different ways, software-wise as well as hardware-wise. The security issue in IoT is certainly gaining more attention presently, and new approaches are being proposed to improve this aspect. For instance, some authors present testbeds to approach the difficult task of assessing security in IoT deployments [7].

In this work, these main concerns related to the security on the Edge of IoT are addressed, by creating a hardware platform that combines a main processing core with a Hardware Security Module in a modular and flexible architecture, so as to foster protection strategies for current and future sensor network deployments. Different techniques are used to guarantee privacy and integrity in the data collected by sensor nodes, as well as mechanisms to join the network in a trustable manner. The design and implementation of the hardware layer have been conceived to produce a trade-off solution between computational performance, power consumption awareness and high degree of protection with dedicated hardware resources, particularly considering the increasingly importance of the active operations of the nodes in IoT dynamic contexts. The runtime self-diagnosis management of the Edge node by providing power and functional islands, real-time current consumption monitoring and an extended range of operational modes for advanced power profiles are key features that this work takes into account to provide dynamic adaptation for the target application scenarios.

Secondly, in order to validate and provide a baseline hardware and software platform for supporting distributed IoT Extreme Edge applications, a lightweight and robust multi-hop communication strategy for the Extreme Edge of IoT is proposed in this work (called Extreme-LT Routing) that allows verifying the dynamic deployment, discovery, data processing and dissemination of IoT devices in a reliable yet low-power resource-constrained fashion. The presented routing algorithm is based on the self-composition of the network topology based on the deployment conditions of the wireless nodes to find the best possible routes for the given circumstances, so as to achieve an optimized data delivery from the sensing nodes to the Edge of the IoT layers. The design and implementation of this technique is included as an embedded software component of the proposed IoT platform, and it has been used as the support communication capability to analyze its behavior and performance in real IoT Extreme Edge deployments, through the realization of intensive experimental tests, as shown in the outcomes of the work.

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

- A modular hardware platform for the edge and extreme edge of the IoT, with HW enhancements for security, trustability and protection against hacking. It includes the implementation of enhanced low-power profiles to provide a trade-off solution between more demanding processing capabilities yet reduced energy consumption.
- An extreme lightweight transmission protocol for multi-hop packet routing in resource-constrained IoT edge sensor networks. Its main features are simplicity, robustness, efficiency and hardware Independence.
- A detailed and extensive set of real experimental tests to study the performance of the proposed solutions on the actual hardware implementation. The enhanced low-power profiles as well as the dynamics of the proposed transmission technique are deeply analyzed.

The rest of the article is organized as follows: Section 2 presents the Cookie modular platform, the particular design and implementation under study in this paper and its main features, as well as the porting and integration of the Contiki-NG operating system into the proposed hardware platform. Section 3 introduces the security aspects of the Internet of Things and their relevance on wireless sensor networks. The implementation of the security solution in the aforementioned platform is also proposed. Then, Section 4 is devoted to detail the lightweight multi-hop communication strategy for dynamic data processing and dissemination on the Extreme Edge, which is intensively tested and validated in Section 4.2, where the experimental results are presented and discussed. Finally, conclusions and future works are highlighted in Section 5.

2. Modular Hardware Platform for the Extreme Edge of IoT

The baseline architecture of the proposed solution for supporting security and distributed applications on the Edge and Extreme Edge relies on a modular hardware platform: The Cookie node [8,9]. This architecture follows a very flexible approach that promotes the implementation of IoT technologies with a very smooth integration effort, by considering the combination and reusability of hardware components in a seamless and modular fashion. The general structure of a Cookie node is composed of four main layers: The processing layer, which integrates the core elements to provide computational capabilities to the sensor node; the communication layer, which includes the wireless technology to provide connectivity to the surrounding network as well as the remote IoT infrastructure; the sensor layer that implements the physical interface to interact with the target environment; and the power supply layer, which is in charge of the voltage level provisioning and debugging capabilities to the rest of the modular platform. The vertical connectors of the Cookie architecture facilitate the plug-and-play philosophy of the platform, which means that new communication, sensing, processing and power supply technologies can easily be integrated without the need to replace or redesign the rest of the layers. Therefore, reusability and adaptability are the main pillars to facilitate fast prototyping upon the hardware architecture [8].

2.1. The Cookie Node

Targeting the provision of security and reliability capabilities on the Extreme Edge, a new IoT hardware platform has been developed in this work following the design style and the modularity of the Cookie architecture. This new self-contained version of the Cookie node aims at the next generation of IoT devices particularly considering key objectives such as trustability, scalability, flexibility and a security-based design, as well as adapting it to the hardware architecture and modularity of the Cookies. In this way, the new Cookie Edge Node is indeed an IoT oriented platform, which includes a Silicon Labs EFR32MG12 SoC as a core of the processing layer and several peripherals for sensing and security purposes. A general schematic view of the Cookie board architecture from a functional point of view is shown in Figure 1. The EFR32 MCU is a 32-bit Cortex-M4 SoC with a maximum operating frequency of 40 MHz, an IEEE 802.15.4 radio and enough memory to run applications with an increased demand on computational resources on the Edge and the Extreme Edge (256 kB RAM, 1 MB Flash). While being a 32-bit chip, it has been designed with the goal of energy efficiency, fast wake-up times and a scalable power amplifier [10]. These features are seized in the Cookie node, while bearing in mind the necessity of establishing a secure and trustable network.

It also includes a SI7021 temperature and relative humidity sensor [11], which can be interfaced via I^{2}C, and an ICM-20648 6-axis inertial sensor [12], which is accessed through SPI.

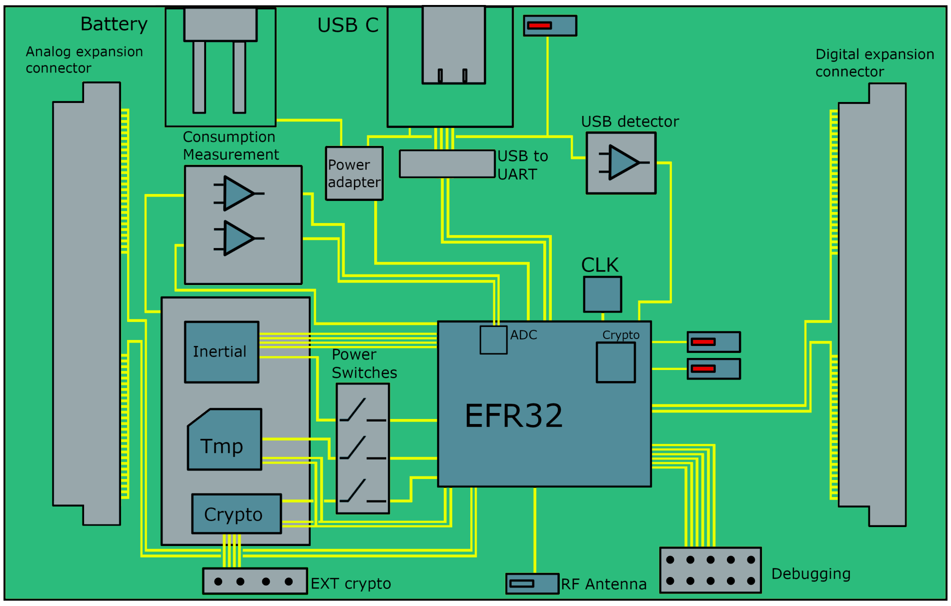


Figure 1. Block Diagram of the designed Cookie Node.

Besides the crypto accelerator integrated in the EFR32, the proposed Cookie node has another cryptographic co-processor. The Microchip ATECC608A encryption chip [13] is a core feature of the board, making it able to run several encryption algorithms and store the secure key on hardware, supporting the establishment of a chain of trust among the nodes in the sensor network. Also, its design makes the chip resistant to side-channel attacks.

With the aim of controlling its energy consumption, the new Cookie layer also includes two operational amplifier blocks at the input of the MCU and the consumption islands. These blocks enable the MCU to measure the consumption of the SoC and the sensors separately, and are directly connected to a 12-bits resolution ADC, therefore allowing the platform to perform self-adapting energy-aware strategies to switch to a better suited power profile, according to the application context needs. The implementation result of the new IoT Cookie node for the Extreme Edge is shown in Figure 2, where a top-layer view of hardware platform is presented.

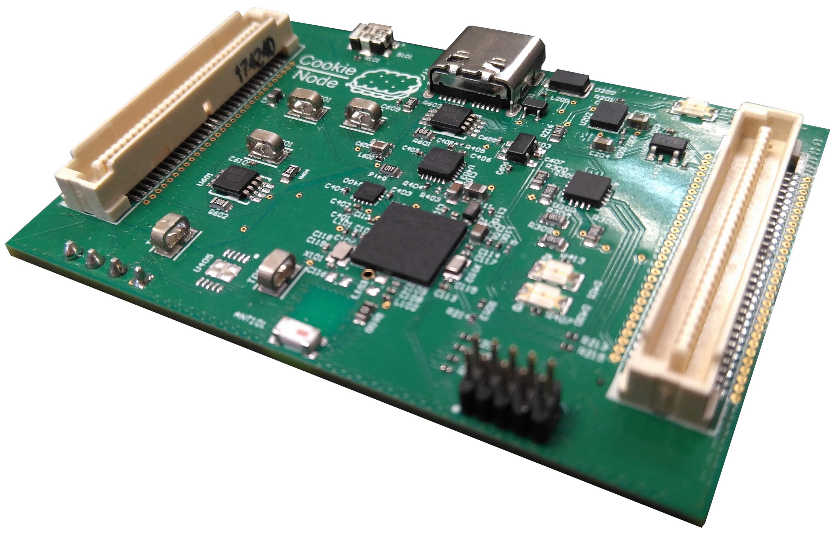


Figure 2. Implementation of the Cookie node with Silicon Labs EFR32 MCU and Microchip ATECC608A Cryptographic Co-Processor.

2.2. Low-Power Strategies

2.2.1. EFR32 Low-Power Modes

According to Hyung-Sin Kim et al. [14], although the idle current of each hardware component is provided by its datasheet, the idle current of a sensor node may be impacted by many additional factors, making it significantly greater than the sum of the datasheet values [14]. The MCU of the Cookie, as part of the EFR32MG12 family, has a variety of low-power modes available. These modes allow the SoC to save energy by reducing the power consumption of the processor when it is not required to display its complete functionality. These modes are depicted as follows [15]:

- EM0 - Active/Run Mode: the normal running mode, everything is active.
- EM1 - Sleep Mode: the CPU clock is disabled. The memory can still be accessed through Direct Memory Access (DMA) and the peripherals can be handled using the Peripheral Reflex System (PRS).
- EM2 - Deep Sleep Mode: not only the CPU clock, but the high frequency oscillators are also disabled. The 32 kHz low frequency oscillator is still enabled.
- EM3 - Stop Mode: all high and low frequency oscillators are disabled except for the ultra-low frequency and, optionally, the auxiliary ones.
- EM4S - Shutoff Mode: all oscillators are disabled, there is no RAM retention and the MCU is shut down except for the recovery logic. The only way to wake up is through an external reset.
- EM4H - Hibernate Mode: similar to the EM4S mode, providing more options for the wake-up call. Mode EM4H can have RTCC running with the ultra-low frequency oscillator, while EM4S cannot. EM4H does also provide some RAM retention, which EM4S does not.

2.2.2. Enhanced Low-Power Profiles

Besides the low-power modes of the EFR32, the designed Cookie node for the Extreme Edge provides software access to enable or disable signals associated with the power supply of the external peripherals, such as the sensors and the cryptochip, then introducing the concept of power and functional islands. In this way, the power consumption of these islands can be arbitrarily controlled and adjusted to the combination that suits best each moment according to the target application and its dynamics. This feature can be combined with the aforementioned energy modes of the processor, therefore improving the consumption saving of the platform and enabling the creation of some more powerful and extended Low-Power Profiles.

All possible combinations of these options for the modes studied in Section 2.4 are highlighted in Table 1.

Table 1. Combined low-power modes of the Cookie Edge Node, allowing the creation of a more advanced range of power profiles.

	EM0	**EM1**	**EM2**
All islands disabled	EM0-000	EM1-000	EM2-000
Temp sensor enabled	EM0-100	EM1-100	EM2-100
Cryptochip enabled	EM0-010	EM1-010	EM2-010
Inertial sensor enabled	EM0-001	EM1-001	EM2-001
Temp + Crypto enabled	EM0-110	EM1-110	EM2-110
Temp + Inertial enabled	EM0-101	EM1-101	EM2-101
Crypto + Inertial enabled	EM0-011	EM1-011	EM2-011
All islands enabled	EM0-111	EM1-111	EM2-111

The naming code for each cell of the table (each combined mode) comes from the combination of the power mode of the processor (EMx) and three bits depending on the on/off state of the power switch of each consumption island, in the following order: the temperature sensor (x__), the encryption chip (_x_) and the inertial sensor (__x). For example, having the EFR32 in normal sleep mode while having the inertial sensor enabled and the other two power islands disabled would be coded as EM1-001.

2.3. *Software Integration and Usability*

To provide additional software support for the proposed Cookie node beyond the embedded libraries developed to use the platform (in case of needed), the Contiki-NG Operating System has been integrated in this hardware node. Contiki-NG started as a fork of Contiki OS [16], with the intent of focusing on the new 32-bits platforms, and the available partial porting of Contiki for EFR32 core [17] has been adapted to the Cookie Edge Node and completed using some of the libraries for the EFR32 from Silicon Labs. In addition to this, the porting has been conceived to provide fully support to the new hardware elements of the proposed solution, including the management of the power and functional islands, the self-diagnosis of the sensor node based on the power consumption monitoring cross-correlated with the advanced power profiles, and the enhanced security capabilities of the Cookie. Moreover, based on the modular architecture of the Cookie platform, and since the vertical connectors have been exploited to make full compatibility of the new hardware design with already existing or future Cookie layers, the different analog and digital signals and their relationship with the connected hardware elements are properly addressed in the implemented porting.

Most of the work to complement and enrich the initial porting and the adaptation to the proposed hardware node can be classified into the following categories:

- Adaptation of the pinout and other purely hardware-related issues, such as mapping ports to new locations.
- Completion of unfinished functions and missing parts of the Software Abstraction Layer (SAL). Since the porting is still a work in progress, some work needed to be done in this area.
- Particularization of generic functions calling to platform-specific ones for each target or board. Since each board provides the user with its own set of Hardware Abstraction Layer (HAL) functions and lower level functions belonging to each MCU, some parts of the higher level needed to be properly connected to the lower layers. This also includes adapting the drivers for the interaction with the peripherals.

2.4. Characterization

From a hardware perspective, this work is heavily focused on sensor nodes for the IoT, and not on the general wireless sensor networks domain. For this reason, it might not cover some of the most popular and traditional devices of the literature and include some others that better fit within the scope of next generation IoT Edge devices, as commented before. Also, it will not compare the proposed platform to smaller 8-bit or 16-bit-based platforms, since the performance and overall purpose of those differ from the aim of the proposed Extreme Edge platform.

To study the different low-power modes, every single combination of the state of the power islands has been tested on the Cookie Edge Node, going through all the energy modes of the processor for each one of them. The main approach was also to test the upper boundaries of the power profile sets in order to provide a trade-off relationship between power consumption and platform performance, seeking a good balance for those more demanding IoT scenarios, as described in the introduction section. The procedure started by initializing the board and forcefully staying in EM0 mode for a few seconds. After that, the MCU went into the next mode (EM1 - sleep) and waited for another 3 s, repeating this process successively from the highest to the lowest power mode. Before going into deep sleep mode (EM2), a low frequency clock was prepared to wake up the MCU and proceed into the next instruction.

During the tests, the peripherals in their enabled state were in idle mode waiting for the instruction to start a sensing cycle, and did not perform any operation or measure. In this way, the measurements obtained are closer to a real behavior since the sensing frequency in a real deployment is supposed to be low, i.e., the purpose of the low-power modes is to save energy when the board is idle, not during a measurement/transmission cycle.

The consumption of each one of the combined low-power modes can be seen in Table 2.

Table 2. Current consumption of the combined low-power modes for the upper active consumption states (mA).

	EM0	**EM1**	**EM2**
All islands disabled	11.93	10.43	6.80
Temp sensor enabled	11.94	10.44	6.81
Cryptochip enabled	11.99	10.45	6.82
Inertial sensor enabled	12.44	10.89	7.33
Temp + Crypto enabled	11.99	10.45	6.83
Temp + Inertial enabled	12.51	10.87	7.34
Crypto + Inertial enabled	12.45	10.95	7.35
All islands enabled	12.52	10.97	7.37

Table 3 shows the approximate current consumption of different platforms in their active state (MCU active, peripherals enabled) and their idle state (MCU sleeping, peripherals turned off). The platforms featured are OpenMoteB (CC2538 SoC based on ARM Cortex-M3, 32-bit), DotNOW emote (STM32F103ZG [18] ARM Cortex-M3, 32-bit) and Sparkfun freeSoC2 [19] (PSoC5LP ARM Cortex-M3, 32-bit). The consumption values shown for the platforms are taken from their respective datasheets, obtained by adding the manufacturer values for the processor consumption in similar circumstances than those of the Cookie for its testing: MCU active with the radio turned off and peripherals enabled but in a wait state. The values for the stm32 consumption (DotNOW emote) that correspond to sleep mode at 36 MHz with all peripherals enabled are shown. The consumption of OpenMoteB was obtained by adding 13 mA of core consumption at 32 MHz and the consumption of some common peripherals: GP timer, USB, SPI, I^2C, UART, but no ADC nor Flash being used. Consumption values for the freeSoC2 were obtained directly from the datasheet for a frequency of 24/48 MHz and 25 °C.

Table 3. Current consumption comparison for the different Edge nodes.

Board (MCU Clock Frequency)	Current
OpenMoteB [20] (32 MHz)	18 mA
DotNOW emote [21] (36 MHz)	17 mA
Sparkfun freeSoC2 [22] (24 MHz)	8.9 mA
Sparkfun freeSoC2 (48 MHz)	15.4 mA
Cookie Layer (38.4 MHz)	12.52 mA

2.5. Hardware Discussion

Figure 3 shows the differences in current consumption between the compared hardware platforms regarding the results shown in Table 3. Since the clock configurations are slightly different for each one of them, the plot has been normalized by considering the outcome of the current consumed per units of MHz. It can be seen that the Cookie layer outperforms the rest of the platforms even with one of the highest clock frequency operations, although in case of the freeSoC2 working at 48 MHz provides quite similar results but for a sleeper state, in contraposition to the Cookie Edge Node in normal mode. In the meantime, the results show that the Cookie layer obtains more than 40% of current consumption reduction in comparison with the OpenMoteB, whose results contribute to optimize the efficiency of the target approach regarding the balance between higher computational duty cycles (thus more presence of active operational modes in the functional profile of the sensor nodes) and power awareness on the Edge.

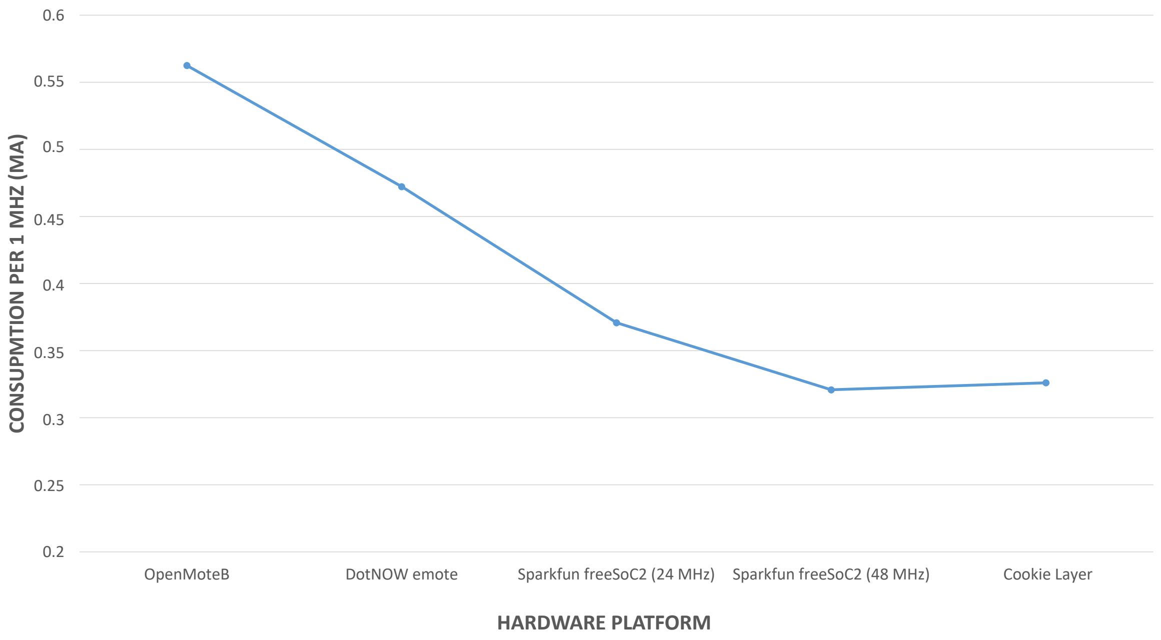


Figure 3. Results comparison of the Extreme Edge nodes considering the current consumption per MHz.

3. Security on the Extreme Edge

Protecting the Edge and particularly the Extreme Edge is one of the main pillars of the proposed modular platform, so as to provide trustability, robustness and reliability in the increasingly complex and diverse application scenarios of IoT. Traditionally, the security issue has been deeply studied in Internet, networking and computer science. However, the ubiquity of IoT devices introduces new elements to the equation, and more vulnerabilities should be considered to be protected from potential attackers. The security schemes are known to have a difficult implementation in real deployments of IoT networks. The operations that take place in common schemes may spend a large amount of time, in comparison with the usual time that a node should be active performing sensing, processing and communication tasks. This is an important issue when the approach for saving energy is to have the nodes in active mode the minimum required time, and move them to a sleep mode whenever possible. Thus, the addition of security capabilities to this type of networks must consider the extra power consumption that will appear.

Moreover, in the new IoT world, it is common to find networks composed of embedded devices that use communication protocols that do not necessarily have access to Internet, as in case of the wireless sensor networks (WSNs), which are oriented to low data rate and low-power consumption. Internet is a network that is continuously being monitored to find irregularities and attacks, but this is certainly not the case of the WSN domain.

The security on the Edge of the IoT is a very serious problem that is being addressed by the scientific community. In this regard, in [23] a security agent is introduced, which is a hardware element with enough resources to carry out advanced security algorithms. This element offloads the security tasks from the restricted sensor nodes, which are working on measuring and sending information to the network wirelessly, although they represent a source of vulnerability, as detailed before. This is the main reason tackling the security and trustability problem directly from the Extreme Edge perspective is gaining important attention.

In this way, one of the main aspects to be considered is that the security should rely on securing the criptographic key, and the ability to keep it hidden from potential attackers, so that a trustable communication between the different parties of the network can be guaranteed. Moreover, side-channel attacks should be foreseen, especially when an attacker may have physical access to deployed nodes. This work focuses on these principles by protecting the key inside the IoT nodes, using dedicated hardware with enhanced capabilities in this regard, with very few overheads in terms of cost and power consumption.

3.1. The Chain of Trust on the Edge

When two members do not know each other, they need to establish a root of trust. This technique is based on the fact that the manufacturer of the equipment or a Certificate Authority (CA) acts as a third member that provides confidence by giving legitimacy to the relationship between the public key and the member who claims to have it.

This process (known as Public Key Infrastructure, or PKI) is a combination of different elements and procedures that allow the execution of the encryption, signature and non-repudiation of transactions or electronic communications using asymmetric cryptography, with guarantees during the whole process. Using PKI, members that do not know each other can authenticate and trust among them before starting a communication. This is done by means of using signatures and certificates. The process consists of the creation of certificates, by the CA, for each device. Subsequently, each member has the public key of the CA with which it is possible to check the validity of the member's certificate with the one a communication (and thus an authentication process) has to be performed. The certificate is a data structure that contains relevant information about the device including its public key, and it is signed by the CA.

This concepts have been brought to the Extreme Edge by the design and implementation of the proposed new Cookie platform, combining the main processing core with a so-called Hardware Security Module (HSM). This dedicated accelerator allows providing the chain of trust with enhanced security capabilities in a transparent and efficient fashion, thus creating a protected modular and trustable hardware node for the Extreme Edge of IoT, as described in the following paragraphs.

3.2. Cookie Node with Enhanced Hardware Security

The Cookie node ensures security and trustability on the Extreme Edge by using the ATECC608A HSM designed by Microchip Inc., which is directly attached to the main I^2C bus, as stated previously in the description of the hardware modules. This chip accomplishes two main tasks. First, the power and time consumption of the cryptographic operations are moved from the software running at the microcontroller to a hardware accelerator, and second, it serves as a trustable module inside the node, meaning that it provides security to store sensitive data inside the platform that will not be discovered by side-channel attacks [24].

The most common strategy adopted when facing the security issue in IoT systems, is the use of symmetric and asymmetric schemes in a mixed fashion, where the authentication processes of the nodes relies on the asymmetric part, and the message exchange is done with symmetric algorithms. These are known to be more efficient if the communication channel is trustable [25]. For asymmetric authentication, usually Elliptic-Curve Cryptography (ECC) is the preferred choice, since the same security level can be achieved with smaller key sizes compared to other alternatives, such as RSA [26]. On the symmetric scheme, the most spread cypher is the Advanced Encryption Standard (AES) with a key length of 128 bits. With this scenario in mind, the ATECC608A was chosen because it provides hardware acceleration for both NIST standard P256 ECC and AES algorithms, and also, the corresponding procedure to switch from asymmetric to symmetric schemes, which is, in this case, the Elliptic-Curve Diffie–Hellman (ECDH) algorithm.

Regarding the capabilities of the HSM to work as an isolated trustable environment, many considerations about its configuration must be done before accessing it from the microcontroller. In order to provide authentication based on a chain of trust, two certificates must be generated prior to the final deployment of the network. The first one identifies the CA, and it is stored in all the HSMs. The second one identifies the HSM itself, and it is signed by the previous CA. Both certificates are stored in a compressed X.509 format in this isolated environment, and must be validated by all the parties involved in the authentication process, prior to verifying the private key associated with the public key included inside the device's certificate. Such a private key is also generated during the configuration stage inside the HSM, and it is never delivered outside the chip under any request. Shared keys for the AES-128 implementation are internally generated by this chip and thus they are never shown.

As already stated, the whole authentication process involves two stages, where the public keys are validated against the signed certificates, and the private key is later checked to be correct. In the first step, certificates are requested to the HSM by the microcontroller, and are exchanged over the network, to perform a validation of the signs and get the public ECC keys of each node. First, the CA's certificate is checked, followed by the device's certificate, where this public key actually resides. The second stage is to verify the private key that is supposed to be the corresponding one to the announced public key. This is done by generating a random number and request for the new node to sign it with its private key, and test the result against the already known public key. All of this is performed with the help of the hardware acceleration provided by the HSM, and the sequence of operations are described in Figure 4.

If the new node succeeds in the verification process, it is labelled as a trustable party. Thus, full communication availability should be allowed. Continuing with the previous ideas, a switch to a symmetric cyphering method is made. The sensor node benefits from the capabilities of the HSM to accelerate the ECDH key exchange that generates a shared secret between the two nodes from the asymmetric key pairs. Since the ECDH algorithm is computed on each node separately, an eventual "authentication confirmed" message should be sent to the new node to coordinate the operation. After both nodes get what is called the pre-master shared secret, a Key Derivation Function (KDF), also available in the HSM, hashes the result one more time. This extra step adds randomness to the previous ECDH operation, and makes the following digest more suitable to use as a symmetric key. A time diagram for this stage is shown in Figure 5.

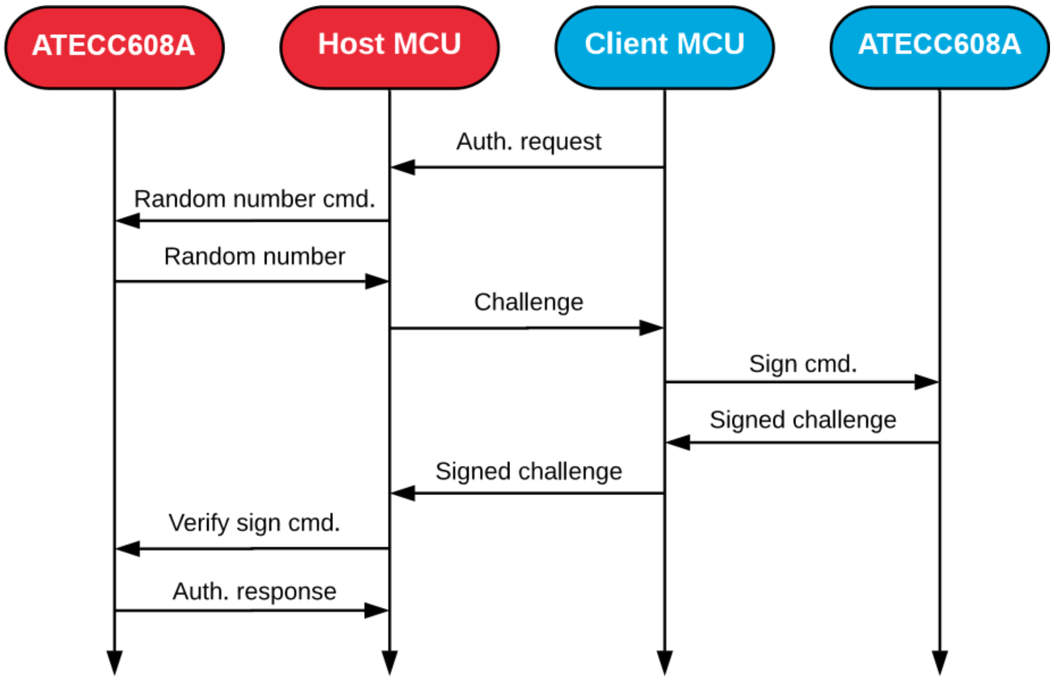


Figure 4. Time diagram of the steps followed in the authentication process between two nodes.

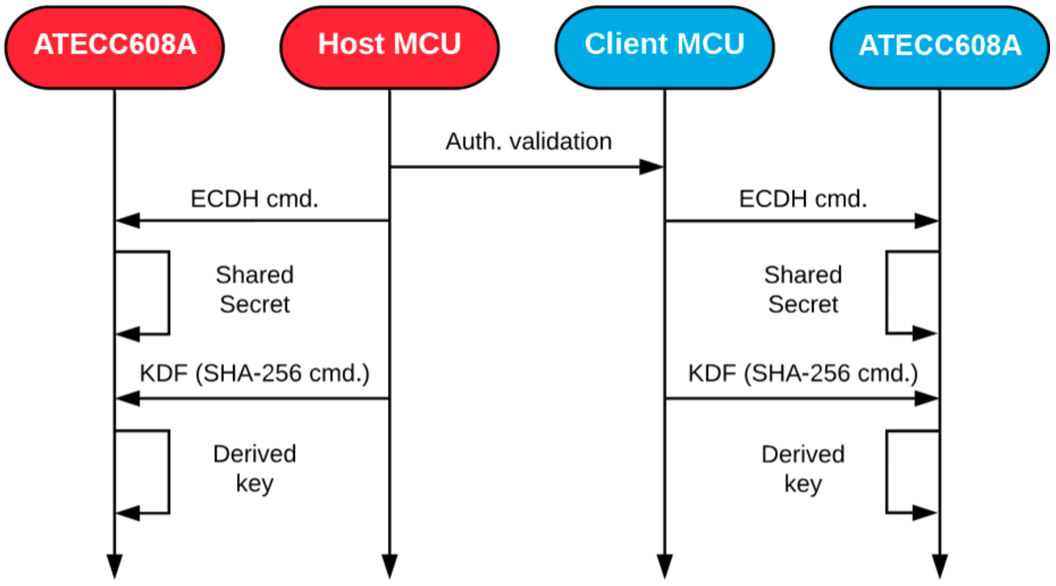


Figure 5. Time diagram of the steps followed to change from asymmetric cryptography to symmetric cryptography.

Once the secure channel for communications has been established, the rest of the messages can be cyphered with the AES-128 algorithm. Notice that the HSM does not support commonly used AES methods of operation, such as AES-CBC (Cypher-block chaining) or AES-CTR (Counter) [27]. Instead, it only provides acceleration of a basic AES engine that works with a single block of 16 bytes. Working only with the engine is not secure because it would not spread the information between different blocks of data, and the resulting cyphertext could not be random enough compared to the original plaintext source. Therefore, it is compulsory for a good performance of the symmetric scheme to coordinate the AES engine of the HSM with an extra help from the microcontroller, to get the behavior of the already mentioned modes of operation.

The main microcontroller of the platform running the software in the sensor nodes also provides its own inner hardware accelerator for AES. Table 4 compares the time spent between this accelerator and the HSM to perform the same encryption tasks, considering different AES modes. Notice that the external HSM provides a better time performance compared to the inner accelerator of the microcontroller, even with the need for additional support from the software part to coordinate the relationships between different blocks (128 bits each one) in the AES-CBC and AES-CTR modes. This reduction in the consumed time, in combination with the low-power characteristics of the HSM, makes this a suitable solution for securing real deployments of IoT edge networks. The added overheads and power consumption is minimum compared to the whole behavior of the network, even with the authentication processes that usually take more time to complete, since those are, in principle, only executed once, and the enhanced security justifies the approach of using a dedicated hardware.

Table 4. Experimental comparison of the two security modules included in the Cookie platform, considering the computational time for different AES modes

	AES-ECB (1 Block)	**AES-CBC (16 Blocks)**	**AES-CTR (16 Blocks)**
EFR32MG12	31 us	210 us	219 us
ATECC608A	7 us	165 us	183 us

Finally, each message exchange should be coupled with a Message Authentication Code (MAC), which allows the destination node to check if there was any error. This MAC can be generated by hashing the message with the SHA-256 function in the HSM. Another alternative would be to take advantage of the Galois field multiplication hardware accelerator of the HSM, which can be used to incorporate the AES-GCM (Galois/Counter Mode) operation [28] to the scheme. This mode calculates and adds the needed authentication code during the cyphering process, saving computational time.

4. The Extreme-LT Routing Protocol

As a means of validating and testing the performance of the Cookie platform in multi-hop distributed deployment contexts, the design and implementation of a dynamic and adaptive routing strategy is proposed in this work, seeking reliability yet lightweight operation for the Extreme Edge. The presented routing algorithm is based on the self-composition of the network topology based on the deployment conditions of the wireless nodes in the target scenario, to find and update the best possible routes for the given circumstances in a lightweight and dynamic fashion, so as to achieve an optimized data delivery from the sensing nodes to the Edge of the IoT layers.

There are some studies in the literature exploring the diverse options for the choice of IoT routing protocols. In [29], the authors focus on ad-hoc routing and study several protocols based on different mechanisms such as distance vector or link state. Another IoT routing protocol is RPL (Routing Protocol for Low-Power and Lossy Networks, [30]), which is widely used and supported by many IoT platforms and operating systems. This is reflected in the existence of many adaptations and variations for it, to enhance its performance in certain scenarios ([31–33]), as well as reviews of RPL-based protocols such as the one in [34].

The Extreme Edge Lightweight Transmission protocol (Extreme-LT) is a lightweight routing protocol developed at the Center of Industrial Electronics (CEI-UPM) for its use on the Cookie modular platform. It is a distance vector routing protocol for IoT networks, focused on simplicity, robustness and reduced processing load. It pursues the goals of reliability, robustness, efficiency and hardware independence set by CTP (Collection Tree Protocol, [35]), adapting and simplifying some mechanisms used by other protocols.

The protocol distinguishes between two node types: sending nodes and a root node. A network will always be composed of a root node, acting as a sink, and a variable number of sending nodes, establishing a tree topology. In this sense, Extreme-LT builds a Destination Oriented Directed Acyclic Graph (DODAG, [30]), similar to the ones used in other IoT routing protocols such as RPL.

The protocol is designed as a simple solution to route messages from the sending nodes to the sink, and since it is conceived to tackle the scenarios where the majority of traffic is directed to the sink node, and having in mind that the environment is lossy and routes are expected to change frequently, there is no necessity to store the whole upstream route in the node using routing tables, as seen in other protocols. Instead, each node only needs to know the route to its parent. Because of this, the choice of the best parent among the candidates is of utmost importance to the establishment of the tree and the optimization of the network topology. For the construction of the DODAG, the protocol uses the rank information and Received Signal Strength Indication (RSSI) as the metric to determine the best parent node from all the potential ones. The procedure to assign the rank of a node follows the following expression:

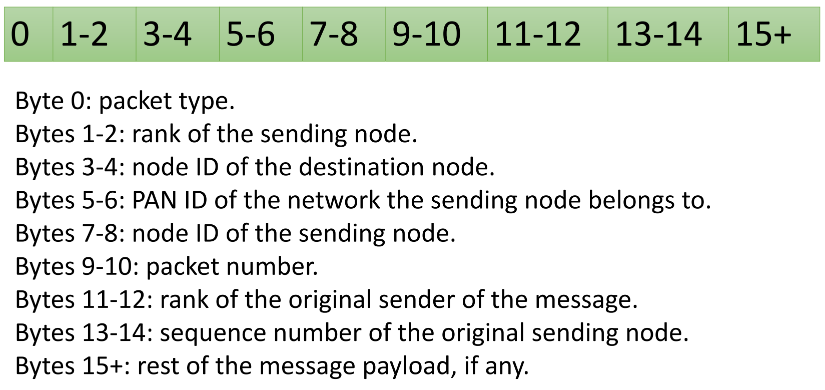
$$Rank(child) = Rank(parent) + HopIncrease \tag{1}$$

With a rank increase per hop of 1 by default, the node rank is equal to the hop count from the root, resulting in the same metric that RPL implements for its Objective Function Zero (OF0, [36]). According to Yassien et al. [37], OF0 is not inferior to MRHOF (Minimum Rank with Hysteresis Objective Function, [38]) in terms of Packet Delivery Ratio (PDR) and power consumption, and even outperforms it in some scenarios. On top of that, Extreme-LT imposes a tie-break policy for equal rank candidates based on their RSSI.

Since a node has no routes stored other than the one pointing to its parent node, for downstream communications the protocol either uses unicast transmissions when it is a response to an upstream message, or uses broadcast messages for the nodes to filter in reception. The former is the usual solution, while the latter is restricted to specific situations to avoid flooding the medium.

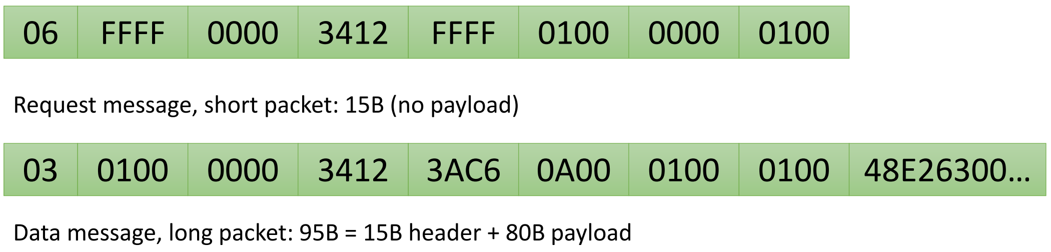
For any given packet being transmitted, the network protocol header frame format includes data from the sender node, such as the node ID, its rank within the network topology or the DAG ID, as well as information related to the packet itself, such as the packet type or the packet number to keep track of the total number of packets sent from a sending node. Different DODAGs, with different DAG IDs, can coexist at the same time.

The protocol relies on the usage of several packet types, ranging from data packets to various kinds of control packets: Request, Discovery (network advertisement), Repair Unicast and Repair Broadcast. These packets have a common header, specific to the protocol, and an optional payload. In particular, data packets have a payload and control packets do not have it. The header frame format of the protocol is shown in Figure 6. To illustrate this frame format, different packet frames can be seen in the examples shown in Figure 7.



Byte 0: packet type.
Bytes 1-2: rank of the sending node.
Bytes 3-4: node ID of the destination node.
Bytes 5-6: PAN ID of the network the sending node belongs to.
Bytes 7-8: node ID of the sending node.
Bytes 9-10: packet number.
Bytes 11-12: rank of the original sender of the message.
Bytes 13-14: sequence number of the original sending node.
Bytes 15+: rest of the message payload, if any.

Figure 6. Extreme-LT header frame format.



06	FFFF	0000	3412	FFFF	0100	0000	0100	

Request message, short packet: 15B (no payload)

03	0100	0000	3412	3AC6	0A00	0100	0100	48E26300...

Data message, long packet: 95B = 15B header + 80B payload

Figure 7. Extreme-LT header examples, with and without payload.

The general functionality of a sending node under the protocol can be seen as a state machine in Figure 8, and Figure 9 shows the corresponding one for the root node. The purpose of the sending nodes is to connect to the network in the best possible conditions, to then start measuring data from the sensors and sending it towards the sink node. For this, a sending node will broadcast a request message when booted. This is the first route creation mechanism provided by the protocol. Any node that receives this message will respond with a unicast discovery message directed to that node. The discovery message contains network advertisement information, including the rank of the node within the network topology. The new node will retrieve the network information from the message and store the node ID as its parent, assigning itself a rank one step higher than the rank of the parent. When other nodes in range also receive the request and send their discovery messages back to the new node, if their rank is better than the current parent or they have equal rank but a higher RSSI, the new node will accept them as its new parent node, replacing the former one. If the rank is lower, the discovery message will be ignored. This mechanism ensures that every node will connect to the reachable parent that offers the best connection to the sink, optimizing the route composition and reducing the number of relay hops as much as possible.

The same request-discovery mechanism triggers when a node loses connection to its parent. When a node encounters a fatal transmission problem at its data link layer, after retrying for a given amount of attempts, the node will delete its parent and broadcast a request, accepting a new parent with the best rank from the nodes within its range. After that, it will broadcast a repair command so that their child nodes can repair themselves, updating their routes and ranks. From the perspective of the rest of the network, this mechanism works as if the node had just been turned on as a new addition to the network, although internally the node will increase its sequence number, so it can track the number of times it has been forced to repair its route.

The second mechanism apart from the request-discovery method, is the network creation from the root. When the root node is booted, it will broadcast a discovery message to advertise the network. Every node in range will connect directly to it, since the sink has rank 0, and spread the network advertisement by broadcasting a discovery message with their own rank. Both mechanisms coexist so the creation of the network can be done in a flexible way, while also making it robust in case of new additions or changes in the topology. An example of a normal startup, depicting both mechanisms, is shown in Figure 10. The flow chart shows a situation where a node A is deployed on its own, with no other nodes nearby to connect to. It will request a rank and receive no answer. After that, the root node is plugged. It will create a network and broadcast a discovery message that node A will receive, accepting the root as its parent node. It will then spread the discovery to other nodes (none in this case). After a while, a node B is connected, and will broadcast a request as node A did. Assuming that the root node is out of range and node A is the only one able to respond to, it will send back a discovery message, being a unicast in this case. Node B will accept node A as its new parent and then it can begin the data transmission towards the network sink through it.

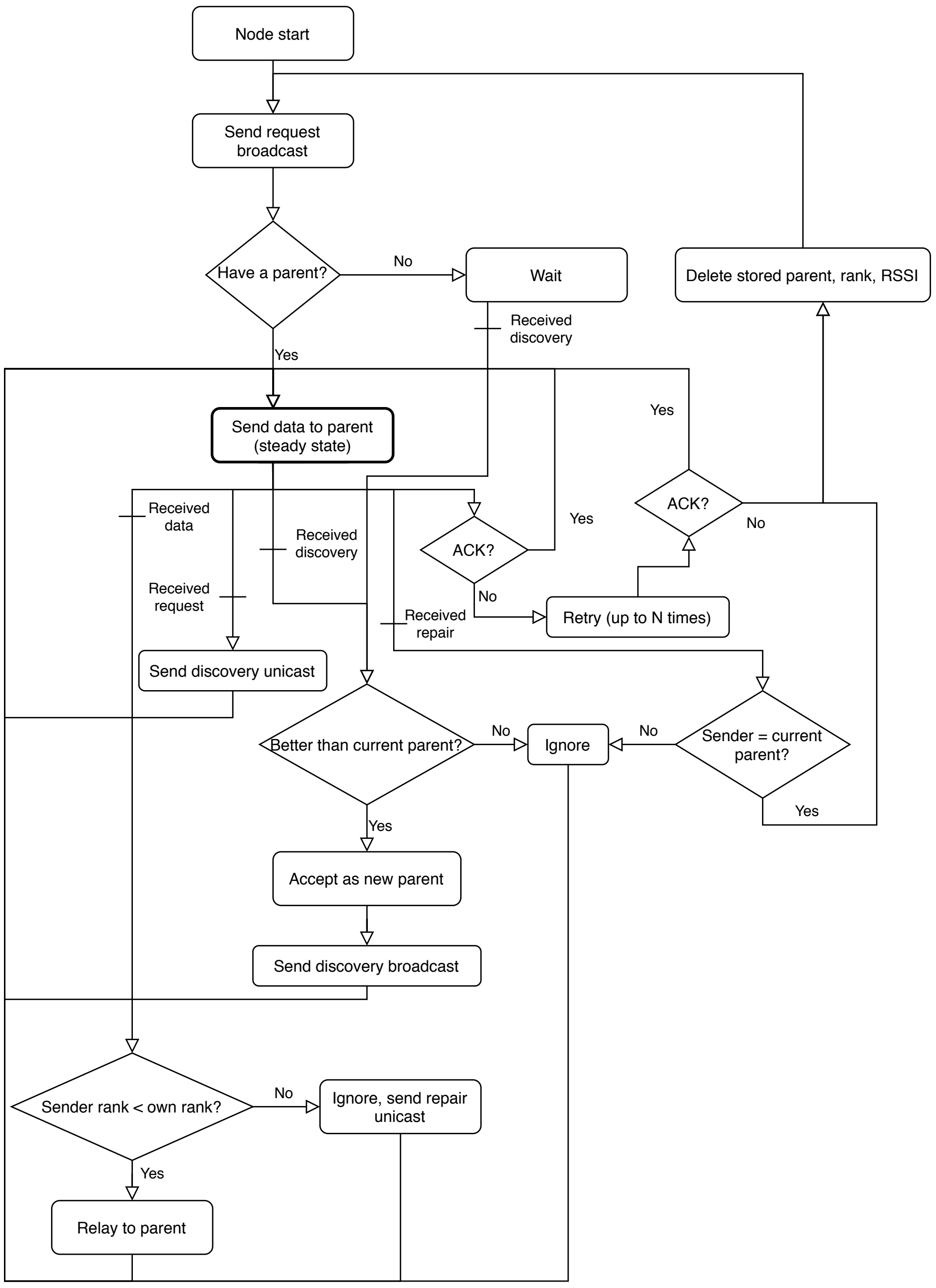


Figure 8. State machine of a sending node (non-root) under Extreme-LT.

After the network is established, the nodes will start sending their data to the sink node. For this, the data packets are always sent to the parent node. The node will first inspect the packet header, checking if the destination is their own node ID or another node ID upstream, i.e., the root ID. It will also check if the rank of the sending node is correct. If it is correct, it will relay the message upstream, or process the content of the payload if the destination was its own node ID. If not, there is an error in the network, since that node should not be sending data to this one. The node will ignore the data packet and send back a unicast repair message.

When a node receives a repair command, it will first filter it compared to its parent node ID. A node will only accept repair commands from its parent. If the sender node ID is the node ID stored as parent, the node will delete it and broadcast a request. After this, it will send broadcast a repair command so its child nodes, if any, will repair themselves and update their routes and ranks. The protocol can be condensed into these two rules:

- A node will only send data packets to its parent node. A data packet from a lower ranked node implies the network topology has changed and needs to be repaired.
- A node will only accept repair commands from its parent node. Repair commands received from any other node will be ignored.

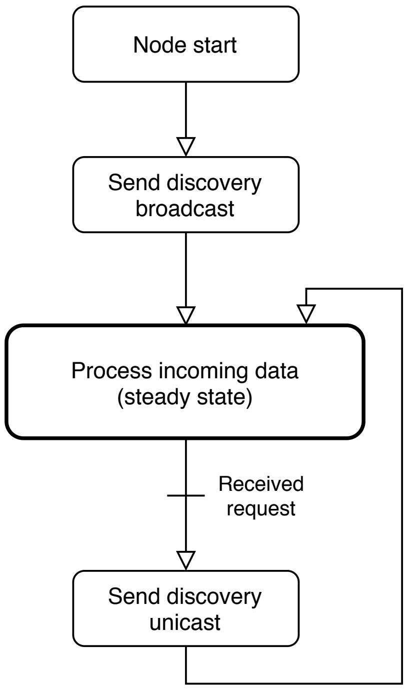


Figure 9. State machine of the sink node (root) under Extreme-LT.

The robustness of the protocol comes from its simplicity. Loops are avoided by ensuring a node will only accept a parent if its rank is the best among all reachable nodes, and will only accept repair commands from the node it has stored as its parent. Discovery messages from nodes with a rank that is not better than the current one will be ignored, and repair commands received from any node that is not its parent will be ignored as well.

As a summary, Extreme-LT is a distance vector protocol developed for the testing and validation of the Cookie platform, but not exclusive to it. It is based on the creation of DODAGs, relying on the robustness granted by the route creation mechanism to implement a reactive maintenance strategy. This way the control packet flow within the network is minimized, reducing the route overheads.

The following section presents the tests carried out on the hardware platform to validate its performance under the protocol, detailing the testing conditions and procedure, the parameters used and the results obtained, which are analyzed and discussed subsequently.

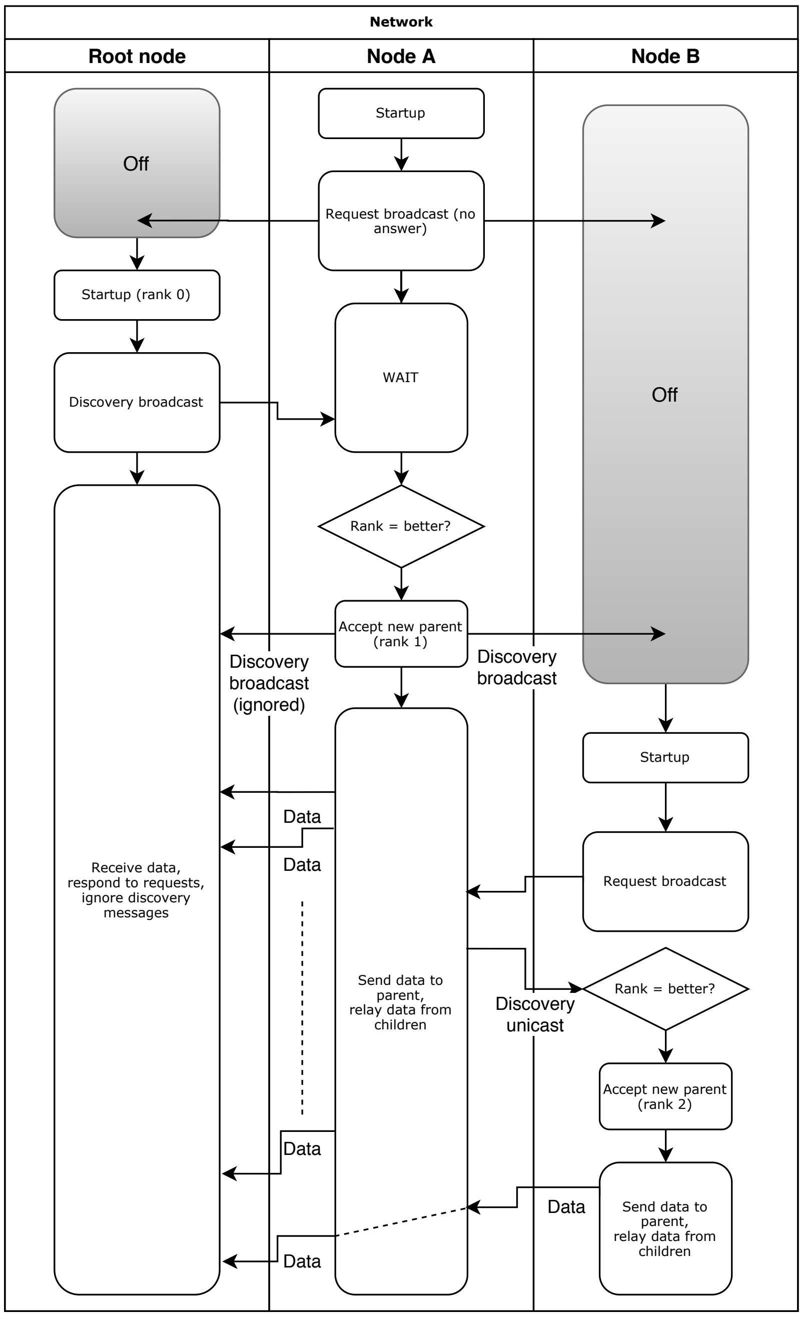

Network
Root node
Node A
Node B
Off
Startup
Request broadcast (no answer)
Off
Startup (rank 0)
WAIT
Discovery broadcast
Rank = better?
Accept new parent (rank 1)
Discovery broadcast (ignored)
Discovery broadcast
Startup
Data
Data
Receive data, respond to requests, ignore discovery messages
Request broadcast
Send data to parent, relay data from children
Discovery unicast
Rank = better?
Accept new parent (rank 2)
Data
Data
Send data to parent, relay data from children
Data

Figure 10. Message exchange in a normal operation under Extreme-LT.

4.1. Preliminary Tests

4.1.1. Range Tests

Before testing the performance of the nodes using the protocol, outdoor tests have been carried out to determine the transmission power of the Cookie platform and the maximum acceptable range of communication. The setup for the tests consisted on a sender node, deployed in a fixed position at ground level, sending packets periodically to a receiver node. The transmission power was set at 20 dBm, which is the maximum power gain of the antenna. The distance between the nodes was initially 1.5 m from which the receiver node was moved away, increasing the separation until the sink node was eventually unable to receive messages. This end condition was met at an approximate distance of over 130 m, beyond that distance the RSSI of the incoming messages was −85 dBm or lower and some of the packets were lost.

4.1.2. Traffic Tests

The next set of tests were designed to create heavy traffic conditions on the sink node and evaluate its capability to receive and process data from several nodes under such conditions. For this, the setup consisted on a receiver node connected to a terminal and 10 sending nodes deployed around it. Transmission power was set at −15 dBm. After being connected, all sender nodes started sending messages to the sink with the following parameters: sending interval = 0.1 s; packet size per transmission = 95 B; minimum number of iterations per node = 1000 (which means that the test lasted until every sending node had sent at least 1000 packets). This sets a worst-case scenario to analyze the performance of the protocol under such conditions. The results obtained are shown in Table 5, computing a total amount of 11041 packets, with PDR (number of packets received at the destination divided by the packets sent by the source, expressed as a percentage) mean equal to 98,4 %. From these outcomes, where the worst PDR was more than 96%, it can be concluded that the sink node is able to endure heavy traffic conditions, being able to receive and process most of the messages sent to it (certainly very close to 100%).

Table 5. Traffic test: saturation of the sink node.

	Node A	**Node B**	**Node C**	**Node D**	**Node E**
Received	1087	1079	1090	1094	1085
Total	1103	1099	1104	1103	1104
PDR (%)	98.606%	98.198%	98.808%	99.203%	98.308%
	Node F	**Node G**	**Node H**	**Node I**	**Node J**
Received	1095	1069	1090	1083	1089
Total	1103	1103	1112	1103	1107
PDR (%)	99.303%	96.1012%	98.24%	98.206%	98.414%

The results of these preliminary tests ensure the sink node will be able to support the incoming traffic and also determine the maximum transmission range of the nodes, and serve as the baseline for the following rounds of testing, in which the performance of the nodes under the routing protocol will be tested.

4.2. Extreme-LT Experimental Evaluation and Results

4.2.1. Setup, Test Procedure and End Conditions

Once the functionality of the platform was verified, a series of tests were performed to trial its behavior under the protocol dynamics, particularly pushing its operation to very extreme boundaries. For these tests, the nodes were deployed in an indoor environment, with the network distribution shown in Figure 11. In this schematic representation of the main lab room (approximately 238 m^2), the red dot represents the root node, acting as a sink, and the yellow dots represent the sensor nodes, able to both generate messages on their own and relay messages from other nodes. The rectangles represent the working disposition of the lab, just as a reference to show the distribution of the nodes and the different locations used during the deployment and testing process.

The setup parameters considered to perform the experimental tests are configured as follows: two packet sizes were used: small packets, with a length equal to 15 B, and large packets, with a length equal to 95 B. These two sizes correspond to those of control and data packets used by the protocol. The message interval for the sender nodes was established at 1 s, 0.5 s and 0.1 s respectively (so very aggressive traffic conditions, in which all the nodes are transmitting and routing packets intensively), with 3 different intervals tested over 2 different packet sizes, for a total of 6 test rounds, where each node generates a minimum of 1000 packets per iteration, as shown below. This setup parameters are summarized in Table 6.

Table 6. Parameters used for each round of testing.

Parameters	Round 1	Round 2	Round 3	Round 4	Round 5	Round 6
Packet size	15 B	15 B	15 B	95 B	95 B	95 B
Sending interval	1 s	0.5 s	0.1 s	1 s	0.5 s	0.1 s

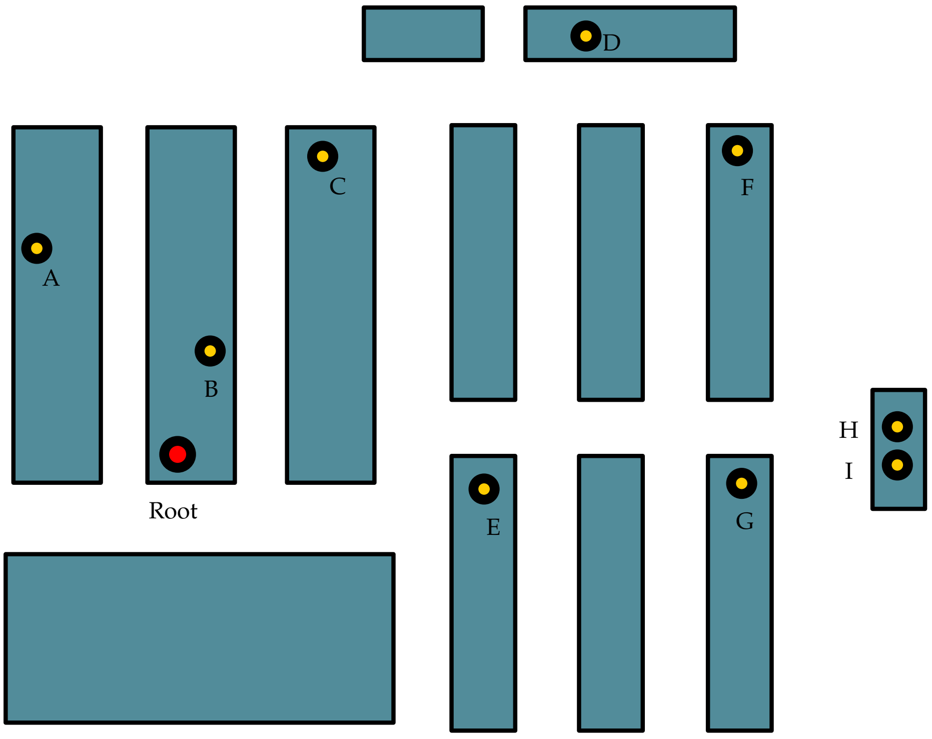
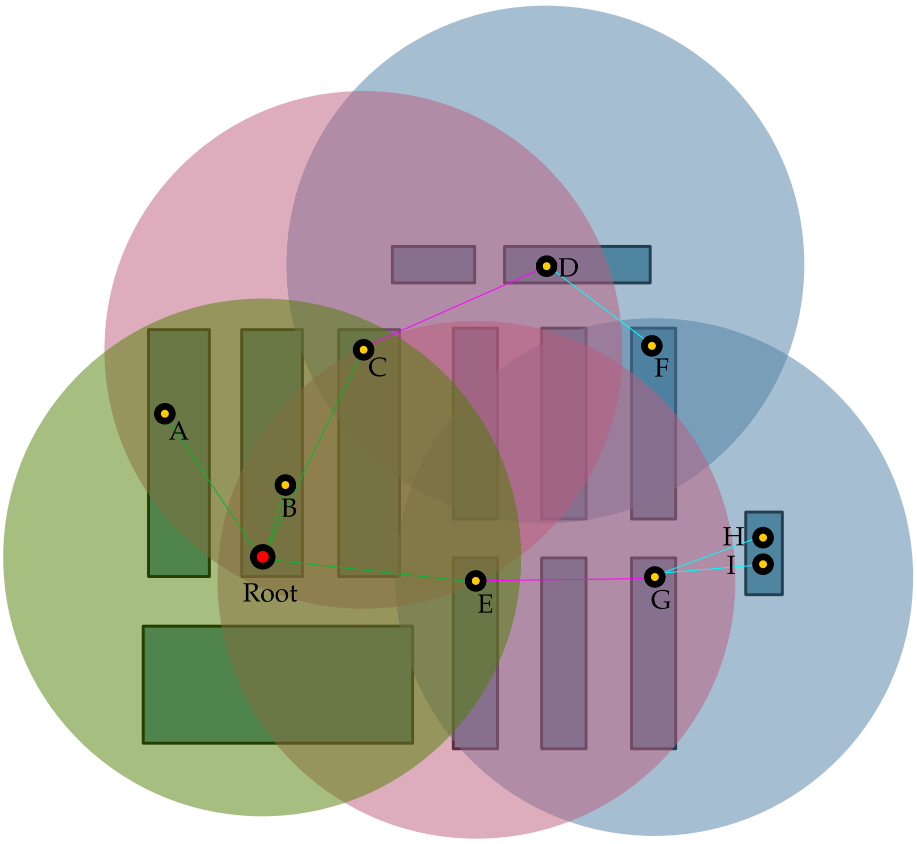


Figure 11. Distribution of the deployed sensor nodes for the routing protocol tests within the indoor scenario.

For each round of testing, the sensor nodes were deployed and turned on in the positions shown in Figure 11, then the root node was connected. By connecting the root node last, the route creation will start from the root node and propagate downstream to the rest of the nodes. This is the second network creation mechanism described in the protocol. An example of the network creation procedure for the setup used in the tests is shown in Figure 12. The topology is established by the network depending on the deployment conditions of the moment. After joining the network, each sensor node started sending packets towards the root node, be it directly or through multiple hops, bouncing in the intermediate sensor nodes. The distribution of the nodes in the lab was the same for all the tests, with their positions fixed, and the connections between them were established automatically according to the protocol, thereby creating some differences in the position of each individual node within the topology.

Because of these differences, the nodes will not be addressed individually but attending to their rank in the network: the sink node has rank 0, the nodes directly sending to it have rank 1, and so on.

Each round of testing was stopped after every node had sent a minimum of 1000 packets to the sink. With this, the PDR of the nodes can be measured and compared, to determine the impact of the packet size and sending interval.



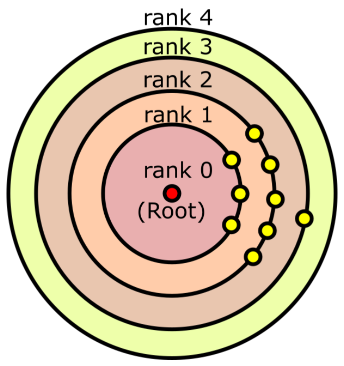
Route creation example:
1) Root connection (rank 0)
2) Root -> A, B, C, E (rank 1)
3) A -> root, B, C (ignored)
B -> root, A, C, E (ignored)
C -> D (rank 2), root, A, B, E (ignored)
E -> G (rank 2), root, B, C (ignored)
4) D -> C, F (rank 3)
G -> E, H, I (rank 3)
5) F -> D, H, I (ignored)
H -> G, F, I (ignored)
I -> G, F, H (ignored)

Figure 12. Example of network creation from the root.

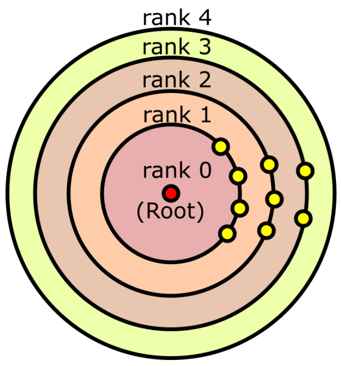
4.2.2. Test results and discussion

The results obtained for each node, sorted by node rank, can be seen in Table 7. For each of the rounds of testing, the route creation mechanism established the network topology, resulting in the node distribution shown in Figure 13a–f.

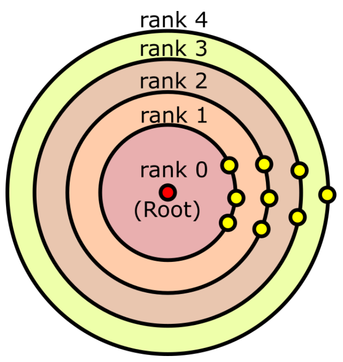
From these results, it can be concluded that as expected, a lower sending interval increases the time the nodes are busy, making a node less capable of relaying messages. This condition makes the routing protocol produce more disperse routes with less child nodes per parent over a highly ramified tree, with many nodes connected to a single node in the same branch. This is, when the saturation of the nodes increases, the protocol tends to form N-ary subtrees with a lower N. In this way, the protocol avoids bottlenecks at route creation, even if it implies that the network will have a higher traffic overall (which ultimately compensates for the possibility of losing packets and/or the number of retransmissions produced by bottlenecks).



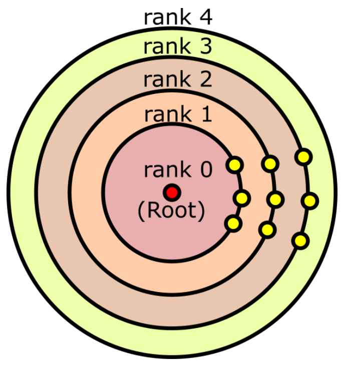
(**a**) Test round 1: 15 B, 1 s.



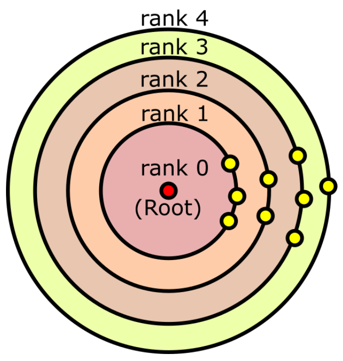
(**b**) Test round 2: 15 B, 0.5 s.



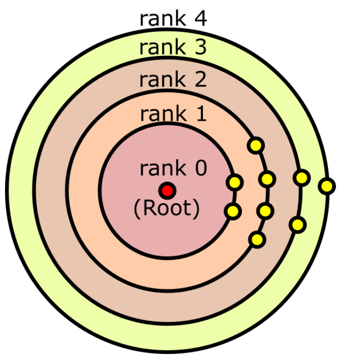
(**c**) Test round 3: 15 B, 0.1 s.



(**d**) Test round 4: 95 B, 1 s.



(**e**) Test round 5: 95 B, 0.5 s.



(**f**) Test round 6: 95 B, 0.1 s.

Figure 13. Node rank distribution generated for rounds 1 to 6.

In these circumstances, a node that could have rank 2 connecting directly to a rank-1 parent node has instead rank 3, because the rank 1 parent node is saturated and does not accept the request from a potential child node (which then connects to a higher rank node that is less saturated). On the other hand, this decision will effectively increase the traffic of the network, since the intermediate rank 1 node will have to route messages coming from all the nodes of its branch, regardless of whether they are sent directly to it or through a relay node.

Also, the congestion comes with a higher packet loss rate. By lowering the sending interval, the PDR from the higher rank nodes drops, reaching rates around 70% in the worst cases. This is due to the higher load put on the relay nodes, since those nodes must send their own messages and redirect messages coming from their child nodes.

For an easier interpretation, the results of each round of tests have been merged, grouping the received and total number of messages attending to the rank of the nodes and obtaining a combined PDR of the nodes with rank 1, rank 2 and so on. These combined results are depicted in Table 8.

A comparison of these results is presented in Figure 14. There is indeed a tendency to avoid bottleneck nodes and disseminate the routes. Such a tendency is accentuated as the sending interval decreases, as can be seen when comparing the results from round 3 to rounds 1 or 2, but there is no direct correlation between the increase in sending frequency and a lower PDR in some cases. As the figure shows for rounds 5 and 6, a faster sending interval (T = 0.1 s) for the same packet size forced the network to route packets in a different way, achieving better delivery rates for each rank than those obtained for a slower message frequency (T = 0.5 s). This is explained by the route creation mechanism. The protocol chooses the best available parent at route creation, selecting the node with the best rank and RSSI as a new parent. Once the route is created and all nodes have started sending, it may occur that the parent node is saturated most of the time due to the high message load from other nodes, being unable to route messages. Thus, the node will delete it and look for a new one, selecting a parent able to correctly receive and route its messages (event with a higher rank) due to having lower load.

In a less saturated scenario, the node might retain its initial parent and stay working under heavier traffic conditions, resulting in a lower PDR. In more saturated circumstances, this initial parent was rejected due to its high load and incapability to relay all the messages it received, resulting in a better PDR due to the reactive mechanism that establishes the routes.

Table 7. Individual node results for each test round.

Round 1	**Node A**	**Node C**	**Node B**	**Node D**	**Node F**	**Node E**	**Node H**	**Node G**	**Node I**
Received	1316	1366	1368	1365	1365	1367	1240	1362	1243
Total	1367	1369	1369	1369	1370	1368	1301	1370	1300
%	96.269%	99.781%	99.927%	99.708%	99.635%	99.927%	95.311%	99.416%	95.615%
Rank	1	1	1	2	2	2	2	2	3
Round 2	**Node A**	**Node C**	**Node E**	**Node B**	**Node F**	**Node G**	**Node D**	**Node I**	**Node H**
Received	1144	1137	1140	1132	1135	1132	1136	1136	1137
Total	1149	1179	1146	1135	1178	1149	1148	1179	1149
%	99.565%	96.438%	99.476%	99.736%	96.350%	98.520%	98.955%	96.353%	98.956%
Rank	1	1	1	1	2	2	2	3	3
Round 3	**Node A**	**Node B**	**Node G**	**Node D**	**Node C**	**Node E**	**Node I**	**Node F**	**Node H**
Received	1324	1339	1299	1172	1325	1280	1219	1189	915
Total	1342	1345	1341	1349	1345	1334	1333	1334	1088
%	98.659%	99.554%	96.868%	86.879%	98.513%	95.952%	91.448%	89.130%	84.099%
Rank	1	1	1	2	2	2	3	3	4
Round 4	**Node A**	**Node C**	**Node B**	**Node E**	**Node F**	**Node G**	**Node D**	**Node I**	**Node H**
Received	1184	1148	1169	1082	1120	1062	1105	1026	1035
Total	1203	1201	1201	1195	1190	1201	1189	1190	1191
%	98.421%	95.587%	97.336%	90.544%	94.118%	88.426%	92.935%	86.218%	86.902%
Rank	1	1	1	2	2	2	3	3	3
Round 5	**Node A**	**Node C**	**Node B**	**Node E**	**Node G**	**Node F**	**Node H**	**Node D**	**Node I**
Received	1322	1209	1369	1096	1044	994	918	930	823
Total	1382	1267	1380	1264	1266	1199	1267	1267	1197
%	95.658%	95.422%	99.203%	86.709%	82.464%	82.902%	72.455%	73.402%	68.755%
Rank	1	1	1	2	2	3	3	3	4
Round 6	**Node C**	**Node B**	**Node D**	**Node I**	**Node A**	**Node E**	**Node H**	**Node G**	**Node F**
Received	3653	4319	1486	764	2960	3326	834	1417	936
Total	3707	4345	1583	809	2983	3397	969	1473	1218
%	98.543%	99.402%	93.872%	94.438%	99.229%	97.910%	86.068%	96.198%	76.847%
Rank	1	1	2	2	2	2	3	3	4

Another aspect that stands out is the relationship between packet size and PDR. It is expected that bigger packet sizes will put a heavier load on the nodes and cause an increment in the time the nodes are busy processing the data, thus reducing the idle time they have left to relay messages from downstream nodes. A decrease in PDR is therefore expected, more noticeable as the number of hops increases. As it can be seen when comparing results obtained from rounds 1 and 4, or rounds 2 and 5, a bigger packet size for the same sending interval results in a worse PDR for nodes of the same rank. Also, it is important to highlight that the created traffic is a very extreme case of a normal application context, so it allowed to analyze experimentally the performance of the multi-hop communication strategy and the designed Cookie node under adverse traffic conditions.

Table 8. Combined PDR attending to node rank.

Test Round		Sum Rank 1	Sum Rank 2	Sum Rank 3	Sum Rank 4
Round 1					
	Received	4050	6699	1243	0
	Total	4105	6778	1300	0
	%	98.660%	98.834%	95.615%	0.000%
	Nodes	3	5	1	0
Round 2					
	Received	4553	3403	2273	0
	Total	4609	3475	2328	0
	%	98.785%	97.928%	97.637%	0.000%
	Nodes	4	3	2	0
Round 3					
	Received	3962	3777	2408	915
	Total	4028	4028	2667	1088
	%	98.361%	93.769%	90.289%	84.099%
	Nodes	3	3	2	1
Round 4					
	Received	3501	3264	3166	0
	Total	3605	3586	3570	0
	%	97.115%	91.021%	88.683%	0.000%
	Nodes	3	3	3	0
Round 5					
	Received	3900	2140	2842	823
	Total	4029	2530	3733	1197
	%	96.798%	84.585%	76.132%	68.755%
	Nodes	3	2	3	1
Round 6					
	Received	7972	8536	2251	936
	Total	8052	8772	2442	1218
	%	99.006%	97.310%	92.179%	76.847%
	Nodes	2	4	2	1

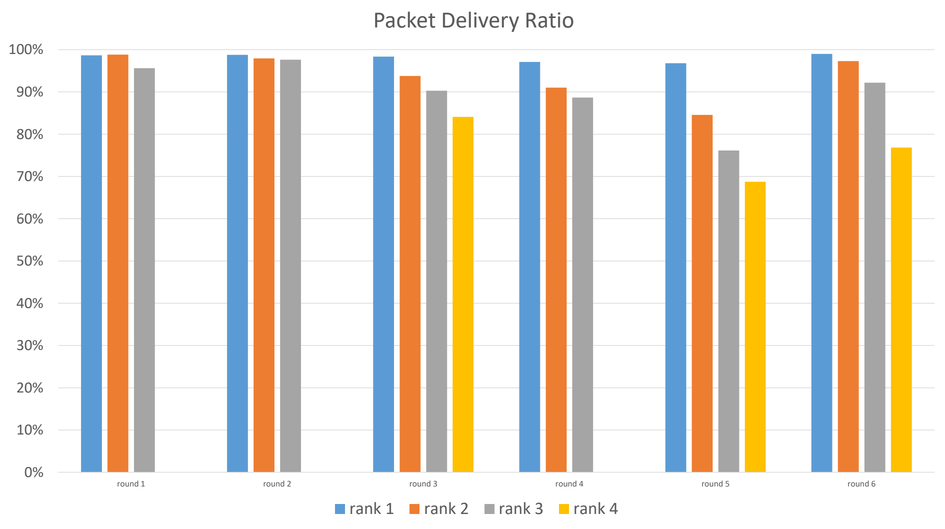


Figure 14. Results comparison of PDR for different packet sizes and sending intervals, grouped by node rank.

5. Conclusions and Future Work

In this work the new version of the Cookie node for the Extreme Edge of IoT is fully presented, a modular hardware platform conceived and designed to provide trustability and robustness necessary for the present and future of IoT applications, and based on the flexibility and scalability paradigm of the Cookie platform. The functionality of this Edge node has been showcased in real experimental performance tests to validate both the hardware and software integration of the proposed system. Additionally, a porting of the Contiki-NG operating system to the platform has been developed as an example of the flexibility and adaptability that is targeted with this new IoT sensor node, which opens the possibility of porting different operating systems into the platform in the future.

Moreover, a lightweight routing protocol designed for sink networks, one of the most commonly used topologies in IoT, is also presented. The protocol takes advantage of mechanisms used by other protocols and implements a simple, lightweight and robust multi-hop communication strategy for the Extreme Edge of IoT. Its performance has been tested on the Cookie platform, obtaining an extensive analysis of the routing mechanism within intensive communication scenarios with heavy traffic patterns, where the amount of data to be transmitted within the network has been overloaded to study the behavior of the sensor nodes within such extreme conditions.

On the other hand, security is a major concern in IoT, and the Edge is the most vulnerable part of the whole ecosystem. A hardware platform with a security-conscious conception during the design and implementation of the proposed Cookie Edge Node solution has been introduced. With very few costs and power consumption overheads, the security increases dramatically. Overall, the results show that a high balance between performance, security and power awareness, and self-diagnosis in dynamic scenarios (where the active operation, participation and collaboration among the nodes is an increasingly common feature in IoT), is certainly possible with the proposed design in this work. In this sense, the proposed Cookie platform is currently being used in practical use cases within the railway field, to provide trustability and chain of trust for on-board and on-track sensor network deployments. The presented platform is serving as the baseline IoT sensor node technology for such application contexts, and further in-field network deployments will be fully supported by this hardware platform.

Author Contributions: Conceptualization, P.M. and G.M.; Data curation, P.M. and G.M.; Formal analysis, P.M, G.M., J.S. and J.P.; Funding acquisition, J.P.; Investigation, P.M., G.M., J.S. and J.P.; Methodology, G.M. and J.P.; Project administration, G.M. and J.P.; Software, P.M. and J.S.; Validation, P.M. and G.M.; Writing—original draft, P.M, and J.S.; Writing—review & editing, G.M. and J.P. All authors have read and agreed to the published version of the manuscript.

Funding: This paper is a result of the SCOTT project (www.scottproject.eu) which has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422. This Joint undertaking receives support from the European Union's Horizon 2020 research and innovation program and several countries such as Austria, Spain, Finland, Ireland, Sweden, Germany, Poland, Portugal, Netherlands, Belgium and Norway..

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Polastre, J.; Szewczyk, R.; Culler, D. Telos: Enabling ultra-low power wireless research. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*; IEEE Press: Piscataway, NJ, USA, 2005; p. 48.
2. Portilla, J.; Mujica, G.; Lee, J.; Riesgo, T. The Extreme Edge at the Bottom of the Internet of Things: A Review. *IEEE Sens. J.* **2019**, *19*, 3179–3190. doi:10.1109/JSEN.2019.2891911. [CrossRef]
3. Trilles Oliver, S.; González-Pérez, A.; Huerta, J. A Comprehensive IoT Node Proposal Using Open Hardware. A Smart Farming Use Case to Monitor Vineyards. *Electronics* **2018**, *7*, 419, doi:10.3390/electronics7120419. [CrossRef]
4. Mujica, G.; Portilla, J. Distributed Reprogramming on the Edge: A New Collaborative Code Dissemination Strategy for IoT. *Electronics* **2019**, *8*, 267, doi:10.3390/electronics8030267. [CrossRef]

5. Ronen, E.; Shamir, A.; Weingarten, A.; O'Flynn, C. IoT Goes Nuclear: Creating a Zigbee Chain Reaction. *IEEE Secur. Priv.* **2018**, *16*, 54–62, doi:10.1109/MSP.2018.1331033. [CrossRef]
6. Garg, H.; Dave, M. Securing IoT Devices and SecurelyConnecting the Dots Using REST API and Middleware. In Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU), Ghaziabad, India, 18–19 April 2019; pp. 1–6, doi:10.1109/IoT-SIU.2019.8777334. [CrossRef]
7. Siboni, S.; Sachidananda, V.; Meidan, Y.; Bohadana, M.; Mathov, Y.; Bhairav, S.; Shabtai, A.; Elovici, Y. Security Testbed for Internet-of-Things Devices. *IEEE Trans. Reliab.* **2019**, *68*, 23–44, doi:10.1109/TR.2018.2864536. [CrossRef]
8. Mujica, G.; Rodriguez-Zurrunero, R.; Wilby, M.R.; Portilla, J.; Rodríguez González, A.B.; Araujo, A.; Riesgo, T.; Vinagre Díaz, J.J. Edge and Fog Computing Platform for Data Fusion of Complex Heterogeneous Sensors. *Sensors* **2018**, *18*, 3630, doi:10.3390/s18113630. [CrossRef] [PubMed]
9. Mujica, G.; Rosello, V.; Portilla, J.; Riesgo, T. Hardware-software integration platform for a WSN testbed based on cookies nodes. In Proceedings of the IECON 2012—38th Annual Conference on IEEE Industrial Electronics Society, Montreal, QC, Canada, 25–28 October 2012; pp. 6013–6018, doi:10.1109/IECON.2012.6389099. [CrossRef]
10. Silicon Labs. *efr32mg12 Datasheet*; Technical Report, 2018. Available online: https://www.silabs.com/documents/public/data-sheets/efr32mg12-datasheet.pdf (accessed on 18 March 2020)
11. Silicon Labs. *SI7021 Datasheet*; Technical Report, 2016. Available online: https://www.silabs.com/documents/public/data-sheets/Si7021-A20.pdf (accessed on 18 March 2020)
12. InvenSense. *ICM-20648 Datasheet*; Technical Report, 2017. Available online: http://www.invensense.com/wp-content/uploads/2017/07/DS-000179-ICM-20648-v1.2-TYP.pdf (accessed on 18 March 2020)
13. Microchip. *ATECC608A Datasheet*; Technical Report, 2017. Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/ATECC608A-CryptoAuthentication-Device-Summary-Data-Sheet-DS40001977B.pdf (accessed on 18 March 2020)
14. Kim, H.S.; Andersen, M.P.; Chen, K.; Kumar, S.; Zhao, W.J.; Ma, K.; Culler, D.E. System Architecture Directions for Post-SoC/32-Bit Networked Sensors. In *Proceedings of the 16th ACM Conference on Embedded Networked Sensor Systems*; Association for Computing Machinery: New York, NY, USA, 2018; p. 264–277, doi:10.1145/3274783.3274839. [CrossRef]
15. Silicon Labs. *AN0007.1: MCU and Wireless SoC Series 1 Energy Modes*; Technical Report, 2017. Available online: https://www.silabs.com/documents/public/application-notes/an0007.1-efr32-efm32-series-1-energymodes.pdf (accessed on 18 March 2020)
16. Duquennoy, S. *More about Contiki-NG*; Technical report, 2017. Available online: https://github.com/contiki-ng/contiki-ng/wiki/More-about-Contiki%E2%80%90NG (accessed on 18 March 2020)
17. Eriksson, J. *Contiki-ng Porting to the EFR32 MCU*; Technical report, 2017. Available online: https://github.com/contiki-ng/contiki-ng/tree/046f283e241e7e73c454cfccc2783775076a3fe4 (accessed on 18 March 2020)
18. STMicroelectronics. *STM32F103ZG Datasheet*; Technical report, 2015. Available online: https://www.st.com/en/microcontrollers-microprocessors/stm32f103zg.html# (accessed on 18 March 2020)
19. Cypress Semiconductor Corporation. *PSoC5LP Datasheet*; Technical report, 2017. Available online: https://cdn.sparkfun.com/assets/e/6/1/8/4/PSoC_5LP_CY8C58LP_Family_DS.pdf (accessed on 18 March 2020)
20. Texas Instruments. *cc2538 Datasheet*; Technical report, 2015. Available online: http://www.ti.com/lit/ds/symlink/cc2538.pdf (accessed on 18 March 2020)
21. Samraksh. *DotNOW Emote Specification Sheet*; Technical report, 2012. Available online: https://samraksh.com/files/products/DotNOW/emote-spec-sheet.pdf (accessed on 18 March 2020)
22. Sparkfun Electronics. *Sparkfun freeSoC2 Retail Page*; Technical report, 2017. Available online: https://www.sparkfun.com/products/13714 (accessed on 18 March 2020)
23. Hsu, R.H.; Lee, J.; Quek, T.Q.S.; Chen, J.C. Reconfigurable Security: Edge-Computing-Based Framework for IoT. *IEEE Netw.* **2018**, *32*, 92–99, doi:10.1109/MNET.2018.1700284. [CrossRef]
24. Maletsky, K. Attack Methods to Steal Digital Secrets. White Paper, Atmel Corporation. 2015. Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8949-CryptoAuth-Attack-Methods-Steal-Digital-Secrets-WhitePaper.pdf (accessed on 18 March 2020)

25. Henriques, M.S.; Vernekar, N.K. Using symmetric and asymmetric cryptography to secure communication between devices in IoT. In Proceedings of the 2017 International Conference on IoT and Application (ICIOT), Nagapattinam, India, 19–20 May 2017; pp. 1–4, doi:10.1109/ICIOTA.2017.8073643. [CrossRef]
26. Maletsky, K. RSA vs ECC Comparison for Embedded Systems. White Paper, Atmel Corporation. 2015. Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8951-CryptoAuth-RSA-ECC-Comparison-Embedded-Systems-WhitePaper.pdf (accessed on 18 March 2020)
27. Dworkin, M. *Recommendation for Block Cipher Modes of Operation*; National Institute of Standards and Technology Special Publication 800-38A 2001 ED; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2005; pp. 1–23, doi:10.6028/NIST.SP.800-38d. [CrossRef]
28. Dworkin, M. *SP 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*; NIST Special Publication; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007; volumn 800, pp. 1–39.
29. Xin, H.; Yang, K. Routing Protocols Analysis for Internet of Things. In Proceedings of the 2015 2nd International Conference on Information Science and Control Engineering, Shanghai, China, 24–26 April 2015; pp. 447–450, doi:10.1109/ICISCE.2015.104. [CrossRef]
30. Winter, T.; Thubert, P.; Brandt, A.; Hui, J.W.; Kelsey, R.; Levis, P.; Pister, K.; Struik, R.; Vasseur, J.P.; Alexander, R.K. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, 2012. Available online: https://tools.ietf.org/html/rfc6550 (accessed on 18 March 2020)
31. Jenschke, T.L.; Papadopoulos, G.Z.; Koutsiamanis, R.A.; Montavont, N. Alternative Parent Selection for Multi-Path RPL Networks. In Proceedings of the 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 15–18 April 2019; pp. 533–538, doi:10.1109/WF-IoT.2019.8767236. [CrossRef]
32. Bhandari, K.S.; Hosen, A.S.; Cho, G.H. CoAR: Congestion-Aware Routing Protocol for Low Power and Lossy Networks for IoT Applications *Sensors* **2018**, *18*, 3838, doi:10.3390/s18113838. [CrossRef] [PubMed]
33. Sheu, J.P.; Hsu, C.X.; Ma, C. A Game Theory Based Congestion Control Protocol for Wireless Personal Area Networks. In Proceedings of the 2015 IEEE 39th Annual Computer Software and Applications Conference, Taichung, Taiwan, 1–5 July 2015; pp. 659–664, doi:10.1109/COMPSAC.2015.21. [CrossRef]
34. Kharrufa, H.; A.A., A.K.H.; Kemp, A.H. RPL-Based Routing Protocols in IoT Applications: A Review. *IEEE Sens. J.* **2019**, *19*, 5962–5967, doi:10.1109/JSEN.2019.2910881. [CrossRef]
35. Gnawali, O.; Fonseca, R.; Jamieson, K.; Moss, D.; Levis, P. Collection Tree Protocol. In *SenSys '09: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*; ACM: Berkeley, CA, USA, 2009; pp. 1–14, doi:10.1145/1644038.1644040. [CrossRef]
36. Thubert, P. Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552, 2012. Available online: https://tools.ietf.org/html/rfc6552 (accessed on 18 March 2020)
37. Pradeska, N.; Widyaman.; Najib, W.; Kusumawardani, S.S. Performance analysis of objective function MRHOF and OF0 in routing protocol RPL IPV6 over low power wireless personal area networks (6LoWPAN). In Proceedings of the 2016 8th International Conference on Information Technology and Electrical Engineering (ICITEE), Yogyakarta, Indonesia, 5–6 Octorber 2016. doi:10.1109/ICITEED.2016.7863270. [CrossRef]
38. Gnawali, O.; Lewis, P. The Minimum Rank with Hysteresis Objective Function. RFC 6719, 2012. Available online: https://tools.ietf.org/html/rfc6719 (accessed on 18 March 2020)

© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).