


An Approach to Remote Update Embedded Systems in the Internet of Things


Cleber S. Peter   [Universidade Federal de Pelotas | cdspeter@inf.ufpel.edu.br]

Lucas Penning  [Universidade Católica de Pelotas | lucas.penning@sou.ucpel.edu.br]

Alexandra Zimpeck  [Universidade Católica de Pelotas | alexandra.zimpeck@ucpel.edu.br]

Felipe Marques  [Universidade Federal de Pelotas | felipem@inf.ufpel.edu.br]

Adenauer Yamin  [Universidade Federal de Pelotas | adenauer.yamin@ucpel.edu.br]

 Post-Graduation Program in Computation, Universidade Federal de Pelotas, R. Gomes Carneiro, 01 - Balsa, Pelotas, RS, 96010-610, Brazil.

Received: 27 December 2022 • **Accepted:** 23 June 2023 • **Published:** 06 November 2023

Abstract There is a growing initiative on the part of regulatory bodies to employ control over firmware emissions destined for Internet of Things (IoT) devices. In this scenario, this paper presents a new approach, called SOTARU, which proposes the use of a consortium Blockchain among embedded system manufacturers as a way to allow access to the update history of devices from multiple manufacturers through a single infrastructure. The security and robustness of the proposal were evaluated with the help of the Common Open Research Emulator (CORE) distributed network emulator. As a result, it was found that SOTARU stands out in terms of security when compared to other approaches proposed in the literature, as well as being functional even in high latency scenarios.

Keywords: Iot, Blockchain, Remote Update, Embedded Systems

1 Introduction

The remote application update performed by embedded systems, commonly called Over-The-Air (OTA), is essential in the context of IoT solutions. After all, correcting failures, updating security protocols, adapting to new operating scenarios, and using continuous integration and delivery (CI/CD) concepts directly depend on the use of OTA strategies [Lopez-Viana *et al.*, 2020].

Despite representing a trend in the development of embedded systems, the use of OTA till now is a challenge for regulatory agents. After all, part of the approval process for these solutions is a detailed audit of the content of the Firmware developed by the manufacturer to ensure that the application does not contain snippets of malicious code with the deliberate aim of harming the consumer [Inmetro, 2022]. However, in the scenario of implementing remote updates of devices, this approach is flawed, since there is no guarantee that the application running on the device reflects the Firmware audited by the regulatory agents.

Furthermore, despite the proposals for OTA enabling the continuous improvement of security measures adopted in embedded systems, these also constitute a vector for exploiting security flaws. After all, an insecure update mechanism is susceptible to different types of Firmware tampering attacks [Bettayeb *et al.*, 2019]. Therefore, as they manage the manufacturer's intellectual property, OTA approaches must ensure confidentiality, integrity, and availability (CIA) at all stages of the update process.

Considering this, in order to propose minimum security requirements to be met by OTA solutions for embedded systems, a working group of the Internet Engineering Task Force (IETF) proposed a specification called Software Updates for

Internet of Things (SUIT) [Moran *et al.*, 2021]. SUIT proposes the use of End-to-End Encryption (E2EE) between the manufacturer and the device to guarantee the integrity and confidentiality requirements of the update, even with the use of a public communication channel. However, the specification proposed by the IETF is characterized by a centralized approach in which a single server is responsible for storing and distributing update files to all devices. Therefore, SUIT does not guarantee the availability of the service since it presents a Single Point of Failure (SPOF) on the update server with the ability to commit to updating thousands or even millions of devices [Choi and Lee, 2020].

Intending to mitigate SPOF, recent works have proposed that the recording of Firmware emissions be carried out in public Blockchains in such a way as to take advantage of the distributed and immutable character of the data stored on the network [Yohan and Lo, 2018; Tsaour *et al.*, 2022; Mtetwa *et al.*, 2022]. However, in these approaches, the dissemination of data is linked to the financial incentive provided by the manufacturer as payment for storing and distributing the updates, that is, the issuance of the update is conditioned to the manufacturer's resources, since the associated cost is volatile and regulated by the market, that forms around the solution.

That said, this paper presents the SOTARU approach that aims to mitigate the SPOF as well as enable monitoring by regulatory agents on Firmware emissions from manufacturers of embedded systems. To this end, the adoption of a consortium Blockchain is proposed as a way to provide a decentralized and highly available infrastructure for storing and distributing Firmware updates to the devices that make up the Internet of Things.

In addition to this introduction, the paper includes the following organization: Section 2 discusses the works related

to the SOTARU approach, Section 3 presents its architecture together with its main components, and Section 4 details the methodology used in its evaluation as well as the results obtained. Finally, Section 5 presents the final considerations and discusses the research sequence.

2 Related Work

The use of Blockchains as a way to provide a distributed and fault-tolerant infrastructure for IoT solutions has been explored in several research fronts [Wang *et al.*, 2018]. It is no different in terms of OTA approaches, as several works have recently adopted this technology as the basis of their proposals.

In [Yohan and Lo, 2018], the authors propose an architecture formed by the Manufacturer Repository, Blockchain nodes and IoT devices. Each manufacturer maintains the confidentiality and integrity of its updates by storing the files in its own repository. The Blockchain only serves as an indexer of updates, storing the history of all updates issued on the network in a distributed way. The author proposed approach meets integrity and confidentiality requirements by not sharing update files. However, centralization on storage does not meet the availability requirement as it results in a SPOF for the architecture and makes it non-fault tolerant of the Manufacturer Repository.

The smart contracts present in the Blockchain of the cryptocurrency Ethereum are exploited in [Tsaour *et al.*, 2022] to ensure that the distribution agents are only remunerated after distributing the update to the devices. Despite partially meeting the listed security requirements, the proposal depends directly on the interest of third parties to establish the decentralized storage and distribution network, an interest which is conditioned to the financial incentive provided by the manufacturer. In addition, this reward system accentuates the low availability of update files that are intended for a reduced number of IoT devices, after all, the Distributor's remuneration depends on the number of updates applied, which encourages the prioritization of updates with a larger audience target.

The authors at [Baza *et al.*, 2018] address the problem of remote updating of automotive systems, whose great differential concerning other IoT devices lies in their mobility. In this approach, the vehicles themselves become Distributors of the updates, having their reward, similarly to [Tsaour *et al.*, 2022], linked to proof of distribution requested by a smart contract published on the Blockchain. The vehicles in this approach allow intercommunication through the formation of opportunistic networks and become excellent alternatives for the storage and decentralized distribution of updates. However, the update of a Target Vehicle is conditioned to the meeting with a Distributor Vehicle, which may delay the application of a critical update and jeopardize the vehicle's operation.

In [Mtetwa *et al.*, 2022] the authors propose a secure and distributed OTA approach also based on Blockchain, but aimed at updating devices connected through Low Power Wide Area Network (LPWAN) networks. In this proposal, the issuer of the update stores in the Blockchain a metadata

file related to the update and negotiates with the application server a window for distribution in Multicast. The approach is restricted to devices connected through LPWAN and, therefore, does not satisfy the expected interoperability premise of IoT solutions. Furthermore, the authors do not discuss mechanisms to grant confidentiality to the process, which makes the proposal susceptible to reverse engineering attacks.

Finally, Table 1 summarizes the discussion of related works by performing a comparative analysis of the type of Blockchain used and the observed security requirements of each of the proposals in relation to SOTARU.

Table 1. Comparative Analysis of Related Works

Id	Paper	Blockchain	CIA
A1	[Yohan and Lo, 2018]	Public	CI
A2	[Tsaour <i>et al.</i> , 2022]	Public	CI
A3	[Baza <i>et al.</i> , 2018]	Consortium	CI
A4	[Mtetwa <i>et al.</i> , 2022]	Public	IA
-	This Work	Consortium	CIA

Despite showing promise, the works listed do not simultaneously guarantee the CIA requirements. Therefore, the following section presents the purpose of this work, whose objective is to fill this research gap.

3 SOTARU Approach: Conception

SOTARU, whose name means Secure Over The Air Remote Update, is intended for the severely heterogeneous environment of IoT middleware, in such a way that the interoperability requirement guides the design process detailed below.

3.1 Agents

The definition of agents with access to the SOTARU approach, illustrated by Figure 1 and discussed below, aims to elucidate the available resources as well as the role of each agent in the update process.

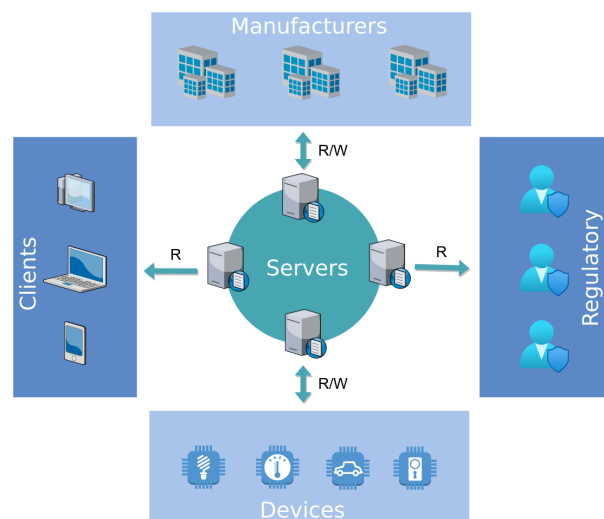


Figure 1. Agents of SOTARU Approach

Manufacturers are the organizations responsible for manufacturing IoT Devices that have a direct interest in updat-

ing their embedded systems. The possibility of remotely updating Devices through server infrastructure presents an opportunity for Manufacturers to better meet their Clients' demands. Furthermore, publicizing the update history makes the updating process more transparent and strengthens the Manufacturer's commitment to Regulatory agencies.

In this proposal, Manufacturers are responsible, through the establishment of a consortium, for maintaining the infrastructure of the update system. To do so, each Manufacturer must provide at least one server for the network that will host, in addition to the update history of all devices produced by consortium members, the update files for Devices that may connect to the server in search of updates. Manufacturers have read and write access to the network, such that writing is performed during the issuance of updates and reading can be used to obtain the update history of a particular device or even to monitor the application of already issued updates.

Regulatory agencies are increasingly taking initiatives to exert control over IoT Devices, especially those considered metrologically relevant according to the Brazilian National Institute of Metrology, Quality, and Technology (INMETRO) INMETRO [2016]. In this sense, one of the approaches adopted by these institutions consists of auditing the firmware content of Devices to ensure that the application developed by the Manufacturer does not have malicious code with the deliberate purpose of harming the Client.

However, this strategy proves to be flawed if there is a possibility of remote Device updates. After all, in this scenario, there is no guarantee that the application running on the Device reflects the firmware audited by the Regulatory agency. Therefore, the proposal of this work provides read access to these institutions on the update history of Devices in order to verify if the firmware issued by the Manufacturer on the network corresponds to the homologated one.

In the SOTARU approach, Devices represent the equipment produced by Manufacturers, used by Clients, and due to their large quantity, are responsible for the need for scaling the solution. Devices are the target of updates and therefore have read and write access to the network. The query process occurs through continuous searching for updates, while writing occurs when notifying the network of the successful application of an update.

Clients represent users of IoT Devices who have an interest in keeping their equipment up-to-date for various reasons, such as providing bug fixes, allowing the addition of new features, and ensuring the continuous improvement of security protocols employed by the Devices. The read access granted to Clients allows them to access the update history of their devices, making the version deployment process more transparent. After all, a resource that becomes viable through the adoption of this proposal is that the Client becomes aware of the modifications that will be applied to their Devices in a specific update, and can even choose not to install a particular version that enables an unwanted feature.

The Server infrastructure, maintained by the Manufacturers who are members of the consortium, represents the central element of this approach and is responsible for the decentralized storage and distribution services of updates.

In the SOTARU proposal, in order to provide agility in response to Device queries, the update history is replicated in

all nodes in the form of a Blockchain using the RAFT consensus algorithm. On the other hand, due to the considerable size of the update files and with the objective of providing scalability to the solution, the files are distributed among the nodes according to demand. In this sense, the RAFT consensus algorithm is also employed to synchronize a Distributed Hash Table (DHT) that maps the location of the files on the network.

The propagation of an update file across the network occurs when a Server receives a query from a Device and verifies in its update history that it is available. At this point, based on its DHT, the node downloads the file from a neighboring Server and makes it available to the requesting Device. Finally, the node updates the DHT to indicate itself as one of the holders of the respective update.

In order to optimize communication between nodes, each server maintains a list of its communication latency with respect to other members of the network. This strategy aims to enable a node, when choosing a neighbor to download an update file, to select the one with the lowest latency and thus reduce the time required for the download, optimizing the use of the communication channel.

3.2 Architecture

The architecture of the SOTARU approach, illustrated by Figure 2, is executed in a distributed way by the servers that make up the infrastructure of the update system and have their modules detailed below.

The Website and Broker modules are responsible for creating the communication interface between the agents and the architecture. The division into two modules aims to facilitate device access through the use of the MQTT protocol, widely adopted in IoT solutions due to its reduced computational cost when compared to HTTP [Yokotani and Sasaki, 2016]. The division, however, also applies to the resources made available, since through the Website the External Agents have access to a series of resources such as issuing and monitoring the distribution of updates, registration and listing of consortium members and their devices, as well as such as access to update history. Devices through the Broker have their access restricted to monitoring and download updates.

The Authentication Module proves to be an essential element for the proposal, acting as a firewall for the architecture, this module regulates access to resources according to the identity and permission level of the requester. In this sense, decision-making regarding granting access is directly related to the requested resource and the Agents' registration data stored in a distributed and decentralized way on the Blockchain. The Blockchain together with the Update Repository represent the persistence modules of the proposal.

The public, sequential and immutable nature of the Blockchain leads to its use for storing temporal events about which there is no interest in changing [Wust and Gervais, 2018]. Therefore, in this proposal, Blockchain is used to store both the device update history and the registration information of the consortium member manufacturers.

The storage of registration data on the Blockchain aims to exploit its decentralized character to provide flexibility to the

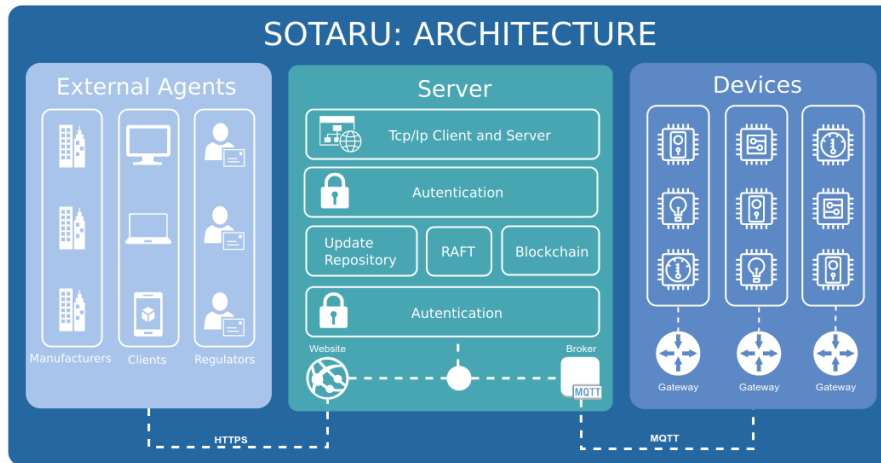


Figure 2. Architecture of SOTARU approach

manufacturer for managing and monitoring its devices. After all, in this scenario the manufacturer can manage its devices not only through its server, but also through connection to any of the network nodes, making communication possible even when its infrastructure is unavailable. The update files, on the other hand, if stored in the Blockchain, would be replicated in all network nodes, reducing the degree of scalability of the solution. That said, in the SOTARU approach, the Update Repository of each member of the consortium only stores the update files of the devices that at some point connected to the server in search of Updates.

The Consensus module refers to the implementation of the RAFT algorithm, which is used in this proposal to ensure synchronization and consistency of the data stored by the network. The RAFT implementation includes an election service, used when selecting a new leader, and a replication and synchronization service, used by the leader to synchronize new records among the other nodes in the network.

As a operational requirement, the proposal must support the distributed and simultaneous generation of records. However, due to the serialization and ordering characteristics of the Blockchain, this proves to be a significant challenge. In this scenario, RAFT provides a potential alternative through the centralization provided by the leader's role.

Finally, the TCP/IP module, composed of a client service and another server, represents the connectivity layer of the approach and allows the interconnection between the member servers of the consortium.

3.3 Consensus Algorithm

This section presents RAFT, used in this proposal as a fault-tolerant consensus algorithm to ensure Blockchain synchronization and homogeneity.

In the state machine applied by the algorithm, the nodes can assume three roles: FOLLOWER, CANDIDATE, or LEADER. Prioritizing consistency at the cost of availability, the LEADER is the node that centralizes the actions for data replication in the FOLLOWER nodes. In its initial state, all servers assume the role of FOLLOWERS and wait for T_L for synchronization messages from the network LEADER.

After T_L , if they do not receive APPEND_ENTRIES messages, the servers enter into a new waiting process, but now

for a random time T_R , that tends to be different for each server. The election process starts when the time $\min(T_R)$ is reached, and the server with the shortest waiting time becomes a CANDIDATE by voting for itself and starting sending REQUEST_VOTE messages to the other members of the consortium.

Each node votes for only one candidate in such a way that the receipt of a number of votes greater than half of the consortium members indicates the end of the election and the definition of the new leader.

Fault tolerance is provided by the election mechanism, which is triggered whenever the LEADER becomes inoperative. With RAFT in a network with N nodes, the consensus is reached with the failure of up to $N/2 - 1$ nodes. That is, the issuance of new registrations is conditioned to the confirmation of at least more than half of the network members.

In the original proposal, interaction with the network is carried out only through the LEADER. However, this centralization limits the network's operating capacity. Therefore, in this work, data reading is provided in a distributed way by the FOLLOWER nodes themselves, while the emission nodes act as a proxy of the connection between the manufacturer and the LEADER.

3.4 Security Strategy

The security strategy adopted in this work, similar to the one presented in Peter et al. [2021], aims to meet the CIA security requirements using E2EE. However, to provide data confidentiality, the manufacturer and its devices must share a common secret, which proves to be a high challenge in the widely distributed context of IoT.

Therefore, in this proposal, we ensure the reliability requirement by utilizing a randomly generated secret key denoted as R_K to encrypt the update content. The encrypted R_K is then shared with devices through a metadata file called Manifest. Additionally, the update's integrity is maintained through digital signature and authentication mechanisms, which rely on distributing pairs of asymmetric cryptographic keys in the following manner:

During a manufacturer’s registration, the Website module generates a pair of public and private keys (P_F, S_F). The publisher’s private key S_F is used to digitally sign the update files and must be kept secret by the publisher. On the other hand, the public key P_F is stored in the blockchain on the devices maintained by this manufacturer and is used for authentication and integrity verification of updates. However, since S_F must remain secret to prevent malicious agents from issuing updates, it was decided as a design criterion for SOTARU not to store private keys on network servers

In the SOTARU proposal, devices that are running the same version of an application are grouped into Projects. This enables updates intended for a particular Project to be directed to all devices that belong to it. The Blockchain can be used as a means of storing and disseminating changes if there is a need to modify the Project to which a device belongs. Each new Project created is also associated with a pair of public and private keys (P_P, S_P). The Project’s private key S_P must be installed on all devices linked to the Project and is used to digitally sign the issuance of notifications sent to the network about the successful execution of an update. In turn, the Project’s public key P_P is stored in the Blockchain and allows both authenticating the messages sent by the devices and establishing, through the Manifest, a secret in common with the servers.

In compliance with the recommendations of the US National Institute of Standards and Technology (NIST) [Barker and Dang, 2015], digital signatures are issued at SOTARU with the ECDSA-SECP256K1 algorithm, in turn, the RSA-2048 algorithm performs the asymmetric encryption of the key RK , and the update content is encrypted with the AES-256 algorithm. In addition, to represent the Manifest and its digital signature in a single message, the JSON Web Signature (JWS) standard is used [Jones et al., 2015].

4 Evaluation

This section presents the evaluation carried out on the proposal of this work. For this purpose, the aspects selected for evaluation are listed below, and the methods and tools used in the process are detailed.

4.1 Security Evaluation

Next, the theoretical behavior analysis of the SOTARU approach is performed when subjected to cyber attacks. The attacks, presented in the form of an attack tree by Figure 3, were selected based on their applicability in the context of OTA approaches [Bettayeb et al., 2019].

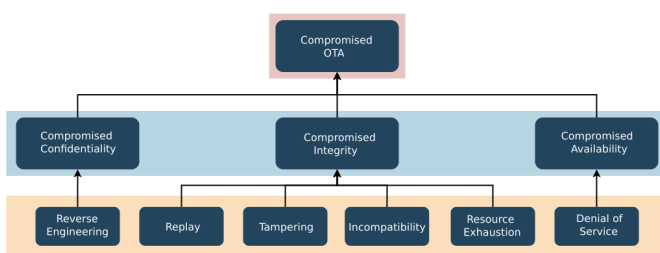


Figure 3. Proposed Attack Tree

In SOTARU, protection against the Replay attack is achieved by using a sequential number that indicates the Firmware version. In this way, the embedded system is only updated to versions higher than the current one.

On the other hand, the guarantee of integrity and authenticity of the update is a result of the digital signature and authentication mechanisms used, which make the proposal safe against Firmware Tampering attacks.

Protection against the Incompatibility attack is guaranteed by the association between the device and the Project, more specifically, the device only executes an update if it has been issued for the Project to which it belongs. Due to the use of the E2EE resource, the proposal also grants confidentiality to the update process, which makes it safe against Reverse Engineering attacks.

The Resource Exhaustion attack, in turn, relies on continually sending rogue updates to the device. In SOTARU, however, the authentication of the Manifest file occurs before the download of the update, mitigating this attack.

Finally, availability and protection against network-targeted DoS attacks are ensured by adopting a distributed, decentralized infrastructure for storing and distributing updates.

Table 2 summarizes the analysis performed comparing SOTARU to the other approaches proposed in the literature. It is possible to verify that the proposal of this work simultaneously meets the three listed security requirements and is safe in relation to the main applicable cyber attacks against OTA approaches.

Table 2. Comparison of Vulnerability between Approaches

Attacks	A1	A2	A3	A4	SOTARU
Replay	●	●	○	○	○
Tampering	○	○	○	○	○
Incompatibility	●	○	○	○	○
Reverse Engineering	○	○	○	●	○
Resource Exhaustion	○	○	●	●	○
Denial of Service	●	●	●	○	○

● Vulnerable ○ Not Vulnerable

4.2 Robustness Evaluation

The robustness assessment of the proposal was carried out with the emulation of the SOTARU approach on a distributed network infrastructure. For that, the CORE [Ahrenholz et al., 2008] emulator was used as a way to explore different network configurations in which the behavior of the proposed approach is evaluated.

The topology adopted in the CORE emulations is defined by the matrix $R = F.[A_{ij}]_{S \times N}$, such that S represents the number of subnetworks, N the number of nodes of each sub-net and $A_{ij} = (i * N) + (j + 1)$.

$$R = \begin{bmatrix} F_1 & \cdots & \cdots & \cdots & F_N \\ F_{N+1} & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ F_{N*(S-1)} & \cdots & \cdots & \cdots & F_{N*S} \end{bmatrix}$$

Considering this, to study the behavior of SOTARU on a larger scale, simulations were conducted for networks without delay, with $S = 4$, and with 32, 64, 128, and 256 servers. The initialization process of these networks was then emulated to record the time taken for the election to complete and the elected node. The results for the network with 32 servers are presented in Figure 4 for the time taken to complete the elections and in Figure 5 for the histogram of the elected servers.

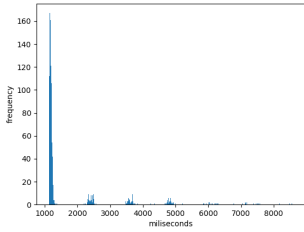


Figure 4. Time of Elections

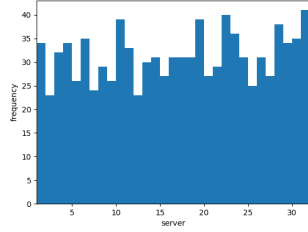


Figure 5. Election - Ideal Network

Based on Figure 4, it's possible to calculate that approximately 80% of the elections held were completed with only one election, while less than 3% required 5 or more elections. Furthermore, Figure 5 shows the random and therefore democratic character of the ideal network election process. After all, the histogram obtained reflects a uniform probability distribution in choosing the leader.

Table 3, displays result of the simulations showing the average time taken for the election process to complete, as well as the Shannon Entropy used to measure the degree of uniformity in the distribution of the elected servers.

Table 3. Performance Analysis on Ideal Network

Servers	Voting Time	Shanon Entropy
32	1.53s	3.46
64	2.34s	4.15
128	5.51s	4.85
256	30.36s	5.54

Table 3 presents that increasing the number of nodes in the network leads to a rise in entropy. According to [Dudewicz and Meulen, 1981], this increment indicates that the distribution's uniformity was maintained, implying that all servers have an equal chance of being elected as leaders. However, it is important to note that the rise in disorder only indicates a greater diversity of elected leaders and does not necessarily mean a decrease in the system's overall efficiency.

Additionally, as shown in Table 3, adding more nodes to the network leads to an exponential increase in voting time. This effect is caused by the competition for the leader position. As the number of nodes increases, the probability of multiple simultaneous candidacies rises, leading to successive votes without consensus. This situation can lead to longer voting times and decreased system performance.

To address this issue, one potential solution is to increase the T_R interval proportionally to the number of nodes. Increasing the waiting interval for the leader reduces the probability of multiple nodes initiating their candidacies simultaneously.

Then, to evaluate the functioning of the approach under more realistic operating conditions, we used latency statistics from the Azure cloud service [Mahesh, 2021] to map heterogeneous and homogeneous distributions of delays in communication between nodes of a network.

In this scenario, the latency matrices are defined as $L = [A_{ij}]_{S \times N}$, where A_{ij} represents the latency in milliseconds of each node based on its position in the network. The matrix of heterogeneous latencies, denoted as L_{he} , is defined by $A_{ij} = (i * 50) + (j * 10)$ and has an average latency of 170ms. Node F32 has the highest latency of 280ms, while node F1 has the lowest latency of 60ms.

To enable comparison between the two scenarios, we define the matrix of homogeneous latencies, L_{ho} , as the average latency of L_{he} , that is, $L_{ho} = AVG(L_{he})$.

$$L_{he} = \begin{bmatrix} 60 & \dots & 130 \\ 110 & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 210 & \dots & 280 \end{bmatrix}, L_{ho} = \begin{bmatrix} 170 & \dots & 170 \\ 170 & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ 170 & \dots & 170 \end{bmatrix} ms$$

Subsequently, emulations of the initialization process for these networks were performed, during which the node selected as the leader was registered. As a result, Figures 6 and 7 show histograms of the servers elected in networks with homogeneous and heterogeneous latency, respectively.

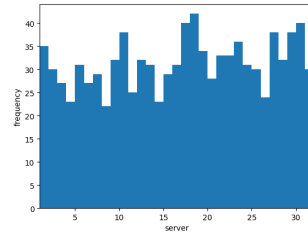


Figure 6. Homogeneous Latency

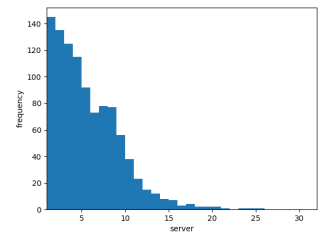


Figure 7. Heterogeneous Latency

Based on the histogram presented in Figure 6, it is possible to demonstrate that the level of democracy in the election process is directly linked to the homogeneity of latency between nodes. In other words, latency does not affect the election process as long as it is consistent throughout the network. On the other hand, Figure 7 displays an Exponential Distribution, which indicates a tendency to choose nodes with the lowest associated latency.

Therefore, it is clear that in a real network, nodes with the lowest associated latency will have a lower average communication latency compared to other network members. Consequently, they will have an advantage in the competition for votes. This characteristic can be advantageous when considering the data synchronization stage since it can facilitate the dissemination of updates. However, the latency imbalance between network members undermines the democratic nature of the electoral process, favoring manufacturers with more computational resources.

Subsequently, to evaluate the proposed fault tolerance mechanism, the behavior of the approach was analyzed in the event of a leader failure. To accomplish this, communication between the nodes was monitored and the leader's failure was emulated. The result of the emulation is shown in the timing diagram of Figure 8.

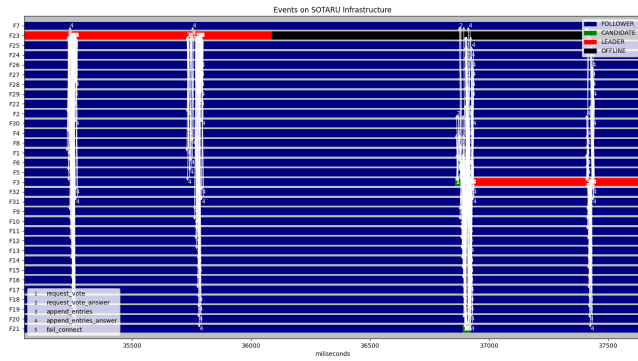


Figure 8. Fault Tolerance Analysis

By analyzing Figure 8, it can be verified that the network is initially in its normal state of operation with node F23 occupying the leader position, a position that is maintained by periodically sending synchronization messages to the other members of the network. Subsequently, the F23 node has its state changed to OFFLINE, which prevents the node from sending and receiving new messages.

The inoperability of the F23 node is detected by the other servers due to the non-receipt of APPEND_ENTRIES messages for a time longer than the T_L timeout. The network manages the leader’s failure with the beginning of a new election process, which in the presented emulation, is contested by nodes F3 and F21 and culminates in the choice of node F3 as the new leader.

4.3 Proof of Concept

To evaluate the interoperability of the proposal, a proof of concept was developed in the nodes of EXEHDA, an adaptive, context-aware middleware based on services that aims to create and manage an ubiquitous environment [Machado et al., 2017].

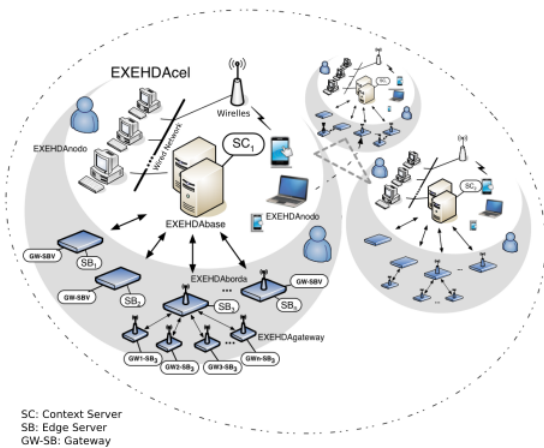


Figure 9. EXEHDA middleware

4.3.1 Device

As a device of the proposed OTA approach, the EXEHDA-gateway node was deployed on the ESP32-DevKitC development board, whose update flow is shown in Figure 10 and described below.

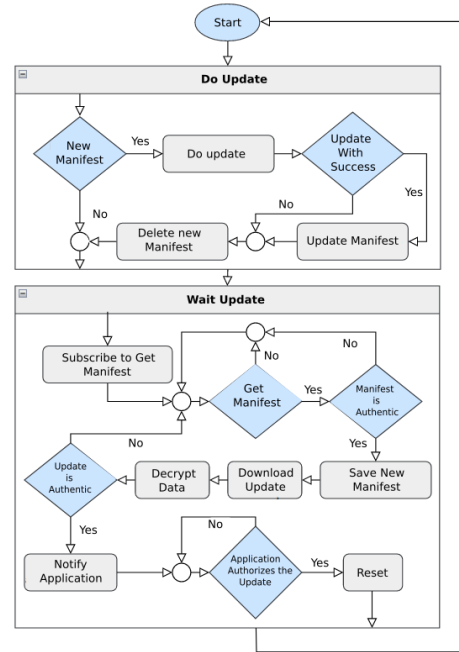


Figure 10. Device Update Flow

The device uses a publish/subscribe strategy to search for updates directed to the Broker component present in one of the consortium member servers. Once it receives the manifest of a new version, the device verifies the authenticity of the Manifest file using the Manufacturer’s public key P_F . Then, it uses the Project’s private key S_P to decrypt the key S_K present in the Manifest to obtain the random session key R_K , which was previously used by the server to encrypt the update contents.

After the successful authentication step, the device requests the server to download the update files. Upon receiving the update files, the device verifies their authenticity using the Manufacturer’s public key P_F . If the authenticity is proven, the device starts the decryption process of the update content through the random session key R_K previously obtained. At this point, based on its operational characteristics, the device determines the most appropriate moment for the reset and subsequent application of the update.

4.3.2 Manufacturer

To compose the proof of concept, four EXEHDAbase node were instantiated to represent the manufacturers. Then, each Manufacturer was associated with a project and an EXEHDAgateway node with the purpose of reflecting the devices to which the updates are intended.

Subsequently, four updates were issued, two for the same Project. With the objective of evaluating the synchronization process carried out by the leader, during the emissions of the last two updates, two manufacturers are disconnected from the network. Consequently, a consensus is reached only among the servers that remained online, generating desynchronization of Blockchain.

Figure 11 shows the server tab that registers the subsequent moment in which one of the servers is reconnected and is in the process of synchronization. This tab allows the user to verify network’s synchronization of Blockchain.

Manufacturer Domain	Status Server	Last Blockchain Block	Status
fabricante.com.br	Online ✔	56gC8cwm6b7jS1QLlR7a7mMUFqix2dYf1R1UxtSXC3oOUYrnyuZBOuG554x56gC8cwm6b7jS1QLlR7a7mMUFqix2dYf1R1UxtSXC3oOUYrnyuZBOuG554x	Synced ✔
fabrica.br	Online ✔	56gC8cwm6b7jS1QLlR7a7mMUFqix2dYf1R1UxtSXC3oOUYrnyuZBOuG554x56gC8cwm6b7jS1QLlR7a7mMUFqix2dYf1R1UxtSXC3oOUYrnyuZBOuG554x	Synced ✔
empresa.br	Offline ✘	M9M8oHy487d0B1KtG6q3k3JeyxQcFdcSchPC2Nk2xa4nBOJ85DFIX2cnTHHbj4206QEqJ27E9Sp7OsfTQ3LTggcoS3OMuEJCJqLBoQgrjMy5HMATgh08	Late 2 blocks ✘
tecnologia.com.br	Online ✔	BtH2kY9a4OWPKm8fS45SmMRtoDupZyM09Kp8FTw7uLeEG68ZMfS1uEidWwSUXt3Fnowka8FEEBeekL478GgcRwypUJHawNMxLcHQeUFR79158	Late 1 block ✘

Figure 11. Network Overview through the Website Servers Tab

5 Conclusions

This work addressed SOTARU, an OTA approach based on resource sharing among embedded manufacturers. The obtained results allow to conclude that the proposal is viable in the widely heterogeneous context of IoT, meeting the required security requirements of OTA approaches and being robust and reliable even when used in high latency networks.

The fault model adopted in the conception of SOTARU allows servers in the consortium to experience inactivity concerning the consensus algorithm used, but it is not intended for them to act maliciously. Therefore, as a direction for future work, we suggest studying consensus algorithms that also support tolerance to Byzantine faults.

Furthermore, as a contribution to the EXEHDA middleware, it is understood that SOTARU represents a strategic effort to advance the use of middleware in interdisciplinary demands involving real-world users.

Acknowledgment

Declarations

Authors' Contributions

All authors contributed to the writing of this article, read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

Data can be made available upon request.

References

- Ahrenholz, J., Danilov, C., Henderson, T. R., and Kim, J. H. (2008). CORE: A real-time network emulator. In *MILCOM 2008 - 2008 IEEE Military Communications Conference*. IEEE. DOI: 10.1109/milcom.2008.4753614.
- Barker, E. B. and Dang, Q. H. (2015). Recommendation for key management part 3: Application-specific key management guidance. National Institute of Standards and Technology. DOI: 10.6028/nist.sp.800-57pt3r1.
- Baza, M., Nabil, M., Lasla, N., Fidan, K., Mahmoud, M., and Abdallah, M. (2018). Blockchain-based firmware update scheme tailored for autonomous vehicles. DOI: 10.1109/WCNC.2019.8885769.
- Bettayeb, M., Nasir, Q., and Talib, M. A. (2019). Firmware update attacks and security for IoT devices. In *Proceedings of the ArabWIC 6th Annual International Conference Research Track on - ArabWIC 2019*. ACM Press. DOI: 10.1145/3333165.3333169.
- Choi, S. and Lee, J.-H. (2020). Blockchain-based distributed firmware update architecture for IoT devices. *IEEE Access*, 8:37518–37525. DOI: 10.1109/access.2020.2975920.
- Dudewicz, E. J. and Meulen, E. C. V. D. (1981). Entropy-based tests of uniformity. *Journal of the American Statistical Association*, 76(376):967–974. DOI: 10.1080/01621459.1981.10477750.
- INMETRO (2016). *Vocabulário internacional de termos de metrologia legal : Portaria INMETRO n. 150 de 29 de março de 2016 / INMETRO*. INMETRO. Available at:<http://www.inmetro.gov.br/legislacao/rtac/pdf/RTAC002399.pdf>.
- Inmetro (2022). *Regulamento Técnico Metroológico Consolidado para Bombas Medidoras de Combustíveis Líquidos: Portaria INMETRO n. 159 de 31 de Março de 2022*. Diário Oficial da União, Brasília, DF, 64 edition. Available at: https://in.gov.br/en/web/dou/-/portaria-n-159-de-31-de-marco-de-2022-*--390396562.
- Jones, M., Bradley, J., and Sakimura, N. (2015). JSON Web Signature (JWS). Number 7515 in Request for Comments. RFC Editor. DOI: 10.17487/RFC7515.
- Lopez-Viana, R., Diaz, J., Diaz, V. H., and Martinez, J.-F. (2020). Continuous delivery of customized SaaS edge applications in highly distributed IoT systems. *IEEE Internet of Things Journal*, 7(10):10189–10199. DOI: 10.1109/jiot.2020.3009633.
- Machado, R., Almeida, R. B., da Rosa, D. Y. L., Lopes, J. L. B., Pernas, A. M., and Yamin, A. C. (2017). EXEHDA-HM: A compositional approach to explore contextual information on hybrid models. *Future Gener. Comput. Syst.*, 73:1–12. DOI: 10.1016/j.future.2017.03.005.
- Mahesh, N. (2021). Azure network round-trip latency statistics. Available at:<https://docs.microsoft.com/en-us/azure/networking/azure-network-latency>.
- Moran, B., Tschofenig, H., Brown, D., and Meriac, M. (2021). A Firmware Update Architecture for Internet of Things. Number 9019 in Request for Comments. RFC Ed-

- itor. DOI: 10.17487/RFC9019.
- Mtetwa, N. S., Tarwireyi, P., Sibeko, C. N., Abu-Mahfouz, A., and Adigun, M. (2022). Blockchain-based security model for LoRaWAN firmware updates. *Journal of Sensor and Actuator Networks*, 11(1):5. DOI: 10.3390/jsan11010005.
- Peter, C. S., Oliveira, T., Monks, E. M., Motta, F. P., Barbosa, J. L. V., and Yamin, A. C. Y. (2021). iota: An approach to secure over-the-air updates on the internet of things scenario. In *Anais do XXVII Simpósio Brasileiro de Sistemas Multimídia e Web*, pages 173–176, Porto Alegre, RS, Brasil. SBC. DOI: 10.1145/3470482.3479630.
- Tsaur, W.-J., Chang, J.-C., and Chen, C.-L. (2022). A highly secure IoT firmware update mechanism using blockchain. *Sensors*, 22(2):530. DOI: 10.3390/s22020530.
- Wang, H., Zheng, Z., Xie, S., Dai, H. N., and Chen, X. (2018). Blockchain challenges and opportunities: a survey. *International Journal of Web and Grid Services*, 14(4):352. DOI: 10.1504/ijwgs.2018.10016848.
- Wust, K. and Gervais, A. (2018). Do you need a blockchain? In *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*. IEEE. DOI: 10.1109/cvcbt.2018.00011.
- Yohan, A. and Lo, N.-W. (2018). An over-the-blockchain firmware update framework for IoT devices. In *2018 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE. DOI: 10.1109/desec.2018.8625164.
- Yokotani, T. and Sasaki, Y. (2016). Comparison with HTTP and MQTT on required network resources for IoT. In *2016 International Conference on Control, Electronics, Renewable Energy and Communications (ICCEREC)*. IEEE. DOI: 10.1109/iccerec.2016.7814989.