Article

A Novel Edge Platform Streamlining Connectivity between Modern Edge Devices and the Cloud

Anderson Carvalho †, Daniel Riordan † and Joseph Walsh *

IMaR Research Centre, Munster Technological University, V92 CX88 Tralee, Ireland; anderson.carvalho@mtu.ie (A.C.); daniel.riordan@mtu.ie (D.R.)
* Correspondence: joseph.walsh@mtu.ie
† These authors contributed equally to this work.

Abstract: This study presents a newly developed edge computing platform designed to enhance connectivity between edge devices and the cloud in the agricultural sector. Addressing the challenge of synchronizing a central database across 850 remote farm locations in various countries, the focus lies on maintaining data integrity and consistency for effective farm management. The incorporation of a new edge device into existing setups has significantly improved computational capabilities for tasks like data synchronization and machine learning. This research highlights the critical role of cloud computing in managing large data volumes, with Amazon Web Services hosting the databases. This paper showcases an integrated architecture combining edge devices, networks, and cloud computing, forming a seamless continuum of services from cloud to edge. This approach proves effective in managing the significant data volumes generated in remote agricultural areas. This paper also introduces the PAIR Mechanism, which is a solution developed in response to the unique challenges of agricultural data management, emphasizing resilience and simplicity in data synchronization between cloud and edge databases. The PAIR Mechanism's potential for robust data management in IoT and cloud environments is explored, offering a novel perspective on synchronization challenges in edge computing.

Keywords: edge computing; cloud connectivity; agricultural data management; IoT in agriculture; data synchronization; cloud computing; pair mechanism; remote farming; data integrity; computational efficiency

check for updates

Citation: Carvalho, A.; Riordan, D.; Walsh, J. A Novel Edge Platform Streamlining Connectivity between Modern Edge Devices and the Cloud. *Future Internet* **2024**, *16*, 111. https://doi.org/10.3390/fi16040111

Received: 7 February 2024
Revised: 15 March 2024
Accepted: 19 March 2024
Published: 25 March 2024

1. Introduction

The global population is expected to increase from just over 7 billion in 2015 to 9.5 billion by 2050, which is a 35% increase [1]. As the earth's population grows it will increase the demand for food. The World Resources Institute [2] acknowledges that an additional 69% in food calories will be required by 2050 to feed the population. Another important factor in terms of food production is the continued movement of people from rural areas to urban areas. To deal with the increasing demand for food and lower availability of workers, farms have had to migrate from manual to automated systems, and are now moving a step further by starting to migrate from being more automated to becoming more autonomous. Modern farms, especially in dairy operations, require robust connectivity and data communications to harness technological advancements fully. This integration is pivotal for real-time monitoring, data analysis, and optimizing farm productivity. Embracing IoT solutions, farms can improve operational efficiency, sustainability, and management, addressing the pressing need to feed a growing population efficiently.

In this study, we embark on an ambitious experiment within the agricultural sector, focusing on the decentralization and synchronization of a central database into 850 distinct databases. These databases are strategically positioned in remote farm locations across 40 different countries. The primary objective of this extensive experiment is to ensure that all these distributed databases remain synchronized, overcoming the geographical and logistical challenges posed by their disparate locations. This synchronization is crucial for maintaining data integrity and consistency across the global agricultural network and is a key factor in effective farm management and data analysis.

In response to the rapid increase in data streaming from edge devices, as observed by [3,4], a new edge device was introduced to each farm's existing setup. This integration was a strategic response to handling more computationally intensive tasks, such as data synchronization and machine learning, more efficiently. As part of this project, a series of experiments were conducted to rigorously test the limits of this newly implemented solution. These experiments ranged from achieving high-speed data transfer, where a 115GB database was offloaded to a broker distributed in 850 distinct queues, in under 10 min, to later experiments that sought a balance between speed and reliability, notably achieving speeds of 124 $\times$ 100 messages per second across a simulated network containing 78 farms.

The relevance of cloud computing in this scenario, despite the lack of a universally accepted definition as discussed by [5], is underscored by its role in storing substantial data volumes. However, as pointed out by [6], the distance between data centers and end-users often poses challenges in low-latency applications and makes the transmission of large data volumes less feasible. In this context, Amazon Web Services (AWS) plays a pivotal role by hosting over 850 databases in several AWS RDS clusters, functioning efficiently under a pay-for-use model.

The combination of the newly added edge device, edge networks, and cloud computing emerges as an architecture that encompasses various layers. This architecture is designed to create a seamless continuum of computing services extending from the cloud to the edge, offering a platform that supports common functions across multiple industries. This horizontal platform is especially beneficial in scenarios like those explained here, where edge computing is central to the research project. Ref. [7] emphasizes that edge computing's primary aim is to bring computation closer to data sources, thus reducing latency, which is a key requirement in modern applications.

The practicality of this approach became evident in the current research scenario, where large volumes of data generated away from urban centers needed efficient processing. This situation highlighted two main challenges: the need to send substantial data amounts from edge to cloud and the necessity to make some cloud-processed information available at the edge within a tight time frame. While theories like those of [8] suggest relying on cloud computing for supplementary resources, this project demonstrates the effectiveness of processing a meaningful subset of data at the edge before transferring it to the cloud, thereby reducing the need for constant data round trips.

In this setup, edge computing, following the insights of [3,4], becomes the nearest point of contact for IoT devices, handling immediate processing needs and only involving the cloud for more complex, storage, or power-intensive tasks. Consequently, edge and cloud computing collectively offer a comprehensive solution for the challenges posed by the increasing volumes of IoT-generated data. While edge computing addresses immediate processing needs with reduced latency, cloud computing continues to manage large-scale data storage and coordination, enhancing overall system efficiency. This integrated approach not only adapts to the burgeoning data demands but also maintains a balanced interplay between localized processing and centralized data management.

The term "edge devices" broadly refers to an assortment of "things" located at the edge of the network, which can pose distinct architectural challenges. As highlighted by [9,10], the extensive scope of these devices—from industrial IoT applications operating within factory confines to expansive smart cities or vast geographic regions—has profound implications for application development [9,10]. These devices can be found in myriad locations, indoor and outdoor, from factories to smart cities or sprawling natural landscapes like forests. The wide distribution of IoT devices, scattered across thousands of different locations, profoundly influences application architecture. For instance, in the current project, the introduction of a new type of edge device proved to be a key element for the development of a novel suite of software and hardware solutions.

Edge devices typically communicate over a heterogeneous network involving various communication channels like Wi-Fi, LTE, and wired connections. This often involves a complex mix of static and dynamic endpoints with varying reachability, which complicates inter-device communication [9]. Moreover, the physical context of devices, such as their location, can significantly impact the application model within fog computing [10].

To illustrate these concepts, consider Table 1, which presents the results of early data transfer experiments between different parts of the system. It demonstrates the time taken to transfer over 5 thousand records, either from cloud to edge, edge to cloud, or from one edge device to another.

Table 1. Recorded communication time by location.

Origin	Destination	Time
Cloud	Edge Solution	<50 ms
Edge Solution	Cloud	<50 ms
Edge Solution	Edge Device	<5 ms

In Table 2, we can observe the average times calculated based on three different attempts to transfer 5 thousand records from the cloud to various edge devices using the proposed edge solution.

Table 2. Recorded communication time by device.

Device	Storage	Time
Raspberry Pi B+©	LiteDB	00:07:35
NVIDIA Jetson Nano©	LiteDB	00:08:15
NVIDIA Jetson TX2©	LiteDB	00:07:55

These results highlight the varying capabilities and performance levels of different edge devices and underline the importance of considering the unique constraints and capabilities of each device when designing and implementing edge computing solutions. The recent surge in data streaming from edge devices, as identified by [4,9], has catalyzed a shift towards processing at the edge, supplementing or even supplanting traditional cloud computing. This concept, often referred to as edge computing or the Mobile Cloud, involves partial processing and analytics on edge devices, with the cloud serving for coordination and data archival, as explored by [11,12].

In this evolving landscape, edge solutions are increasingly empowering industrial plants by enhancing computational capacity at the edge, often through devices equipped with CPUs and sometimes GPUs. This enables existing machinery to utilize machine learning and artificial intelligence, transforming them into smart machines. A pivotal application of this is seen in Deep Neural Networks (DNNs), which, as [13] points out, surpass traditional machine learning techniques in tasks like image classification and speech recognition. Sharma further observes that edge applications are integrating AI for tasks such as augmented reality and face recognition. The study by Thakkar et al. reinforces these points by highlighting the transformative potential of edge AI across sectors such as manufacturing, transportation, and healthcare, by enabling real-time data processing, thus reducing latency and enhancing privacy and security. This underscores the synergy between edge computing and AI in driving forward industrial innovation and efficiency [14].

Yet, the necessity of edge computing alongside cloud computing is debated.Ref. [15] argues that many IoT applications demand real-time data analysis and decision-making, which are requirements often unmet by cloud computing solutions in latency-sensitive contexts. They suggest that AI, in conjunction with edge devices, has the potential to enhance the capabilities of such applications significantly. Thakkar et al. further elaborate on this point by discussing how edge computing, particularly through technologies like Mobile Edge Computing (MEC) and serverless computing, can mitigate the limitations of cloud computing by bringing computational resources closer to the data source. This architectural shift is essential for applications requiring real-time processing, showcasing the complementary roles of edge and cloud computing in modern IT infrastructure. The integration of AI at the edge further amplifies this capability, making a compelling case for the combined use of edge computing and AI in meeting the demands of next-generation applications [14].

The synergy between edge and cloud computing forms a stratified approach to data processing in the IoT era, each offering unique benefits in terms of efficiency, latency, and storage capacity. Understanding their strengths and limitations is essential for their effective deployment within an IoT ecosystem. The introduction of distributed deep learning (DL) techniques, particularly those leveraging Mobile Edge Computing (MEC), presents a promising solution to overcome the computational challenges and enhance security across IoT networks. By offloading computing operations to the edge, these methods support low latency and high accuracy in data processing and analysis, which are crucial for the efficiency of IoT devices with limited computational capabilities.

Edge computing applications are diverse, notably in sectors like smart factories and agriculture. These applications embed software solutions in computing-powered devices at the network's edge to perform various tasks, including sensing, decision-making, and learning. The integration of artificial intelligence across networks of interconnected devices in systems like smart homes, factories, and cities is increasingly addressing complex challenges, from supply chain management to climate change. The evolution towards distributed DL with MEC, introduced by [16] through the BlockDeepEdge framework, represents a significant leap forward in securing and optimizing IoT devices. By facilitating secure, decentralized, and efficient DL operations, BlockDeepEdge addresses privacy and latency issues, enhancing real-time processing and decision-making capabilities at the network's edge. However, while BlockDeepEdge's methodology of distributed deep learning and blockchain provides robust data security and efficient processing, its decentralized nature may not fully cater to scenarios where seamless integration and synchronous operation between cloud and edge environments are crucial. In complex agricultural systems, where the synergy between cloud and edge computing is imperative for comprehensive data analysis and management, the requirement for a harmonized cloud–edge interaction highlights a critical consideration. Although BlockDeepEdge offers significant benefits in terms of security and localized decision-making, its application might be nuanced in contexts demanding intricate cloud–edge collaboration for optimal performance and data coherence across the IoT spectrum.

These data, crucial for optimizing farm management and decision-making, as noted by [17], face the challenge of integration and effective use. The necessity for systematic evaluations, emphasized by [18], underlines the importance of leveraging technologies like Smart Farming and IoT. These technologies facilitate data-driven management to tackle complex challenges in agriculture, from enhancing crop yields to promoting sustainability.

In agriculture, the influx of data from computational devices at the edge is transforming the sector. As the authors of [17] note, the challenge lies in integrating these data to enhance farm management and decision-making. Ref. [18] emphasizes the need for systematic evaluations in promoting sustainable farm systems, highlighting the role of Smart Farming and IoT in facilitating data-driven management.

Real-time reconfiguration capabilities are crucial in agriculture for quick responses to changes like weather fluctuations or disease alerts. This involves enhancing traditional tools with autonomous context-awareness and remote-control capabilities, as discussed by [18]. The integration of robots and human analysis in a cyber–physical cycle is key to this evolution, as outlined by [10].

Refs. [2,19] highlight the use of IoT in monitoring animal welfare, with technologies like Reproductive Management Systems improving livestock productivity. However, cloud-based solutions face challenges like connectivity and bandwidth, especially in remote farming areas.

Edge computing emerges as a solution, reducing latency and bandwidth usage, and enhancing data privacy and sensor battery life. It allows for real-time decision-making, as exemplified in crop irrigation scenarios where sensors process data locally for immediate irrigation decisions. Cooperative frameworks like CoopEdge, introduced by Yuan et al. [20], leverage blockchain for secure and decentralized operations, while architectures proposed by Ju et al. [21] ensure task reliability through comprehensive monitoring. Additionally, innovative data services discussed by Xie et al. [22] facilitate efficient data handling in cooperative edge environments, underscoring edge computing's potential in real-time IoT applications.

To manage the transport layer effectively, MQTT, a lightweight protocol ideal for machine-to-machine communication in IoT, was adopted, as described by [23,24]. Ref. [25] provides a qualitative comparison with CoAP, suggesting MQTT's suitability for applications requiring advanced functionalities like message persistence and secure multicast.

In summary, the interplay of edge and cloud computing, along with efficient transport protocols like MQTT, is reshaping data processing in industries such as agriculture, enhancing efficiency and enabling smarter, sustainable practices.

2. Designing a Distributed Edge Architecture for Remote Agricultural Data Management: Conceptualization and Preliminary Assessments

During the design phase detailed below, extensive research was undertaken to evaluate existing commercial solutions for database synchronization. Notable vendors in this space, such as Oracle with their MySQL offerings, along with others like IBM and Microsoft, provide robust systems designed for database management and synchronization. The investigation also covered a breadth of academic literature on cloud-to-edge database synchronization, with contributions from researchers like Buyya et al. [26] and Grozev & Buyya [27], whose work delves into the intricacies of distributed computing environments. However, none of the existing solutions or academic frameworks were fully compatible with the specific requirements of our project. This necessitated the development of a bespoke solution, one that would seamlessly align with the operational needs of extensive farming, unreliable networks, and the particularities of their data flow processes. The bespoke nature of this solution ensured a tailored fit to the unique challenges posed by the edge computing demands of agricultural technology.

The increasing demand for user-centric IoT applications, which require low latency and real-time data analysis, is a driving force behind this architectural shift. As highlighted by [15], these applications often suffer in cloud computing scenarios due to inherent latency issues. Our proposed edge architecture, therefore, aims to mitigate these challenges, offering a more efficient and responsive alternative.

The scope of our preliminary assessments included a diverse range of farms, from smaller operations with over 60 IoT devices to large-scale farms with up to 6000 IoT devices. Just to give the reader some context, these farms are spread across 40 different countries, quite often in remote locations, providing a rich and varied dataset that enhances the accuracy and applicability of our findings. The insights gained from these preliminary experiments are instrumental in shaping the design of the new edge platform, which aims to alleviate the heavy workload currently borne by cloud applications. By reducing latency and enabling new edge-based services, our architecture promises to revolutionize data management in farming.

In summary, this sets the stage for the in-depth experimental analysis presented later in this article. It provides a detailed account of the journey from concept to preliminary implementation, offering a glimpse into the potential of distributed edge computing in transforming the farming industry.

During the experiments detailed below, a variety of hardware configurations were investigated to define some characteristics needed for an ideal edge device. Configurations used include the Raspberry Pi B+, Nvidia Jetson Nano, Nvidia TX2, Nvidia Xavier, and Intel Nuke, and to simulate 70 farms for a period of over two months, a Fusion Server Pro G5500 Data Center Heterogeneous Server was used.

The development of the architecture presented below is the result of a comprehensive research process initiated approximately two years prior. This process was characterized by an iterative approach, combining theoretical investigation with empirical experimentation, and was significantly influenced by our collaboration with an industrial partner.

Collaboration with senior software and hardware engineers from existing companies played a pivotal role in this developmental phase. This interaction not only facilitated a practical perspective on our theoretical models but also ensured that the developed solutions were congruent with real-world industrial requirements. The direct engagement at some farm's facilities provided an enriched understanding of the operational context, which was instrumental in guiding our architectural decisions.

The learnings from these early experiments, combined with the insights from academic and industrial collaborations, were fundamental in the iterative refinement of our approach. This led to the development of a sophisticated, efficient, and contextually apt edge computing architecture. The following will provide an in-depth view of the architecture that was shaped over these years, detailing its evolution, key features, and the rationale behind its design.

In the early stages of our research, a significant emphasis was placed on developing a prototype for a smart edge computing solution. This solution aimed to bridge the gap between centralized cloud infrastructures and on-site industrial machinery, circumventing the need for substantial investment in new hardware. Figure 1 illustrates the proposed edge solution architecture, delineating the integration of legacy systems with IoT-enabled enhancements.

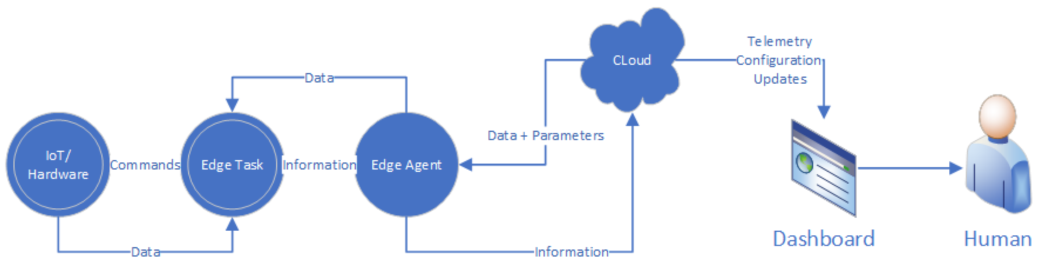


Figure 1. Edge solution architecture.

Figure 1 portrays a simplified cloud-and-edge system, which seamlessly brings together various components and technologies across a range of hardware platforms. The architecture is organized into several layers, with the foundational layer being the "IoT/Hardware" node, indicating the existing industrial equipment. The term "IoT" in this context refers to the potential for augmenting this equipment with IoT add-ons, thereby enabling smarter functionalities.

The "Edge Agent" in the diagram acts as a mediator, facilitating data exchange between the IoT devices and the cloud. It serves a dual purpose: it acts as a local repository and processing unit for data, and as a communication hub that relays information to the cloud or receives instructions back from it. This local processing capability is crucial for reducing latency and allowing for real-time decision-making at the edge.

Transitioning to Figure 2, we elaborate on a more advanced edge computing architecture designed to handle a complex network of thousands of IoT devices, while interfacing with a centralized cloud repository for storage and further data processing.

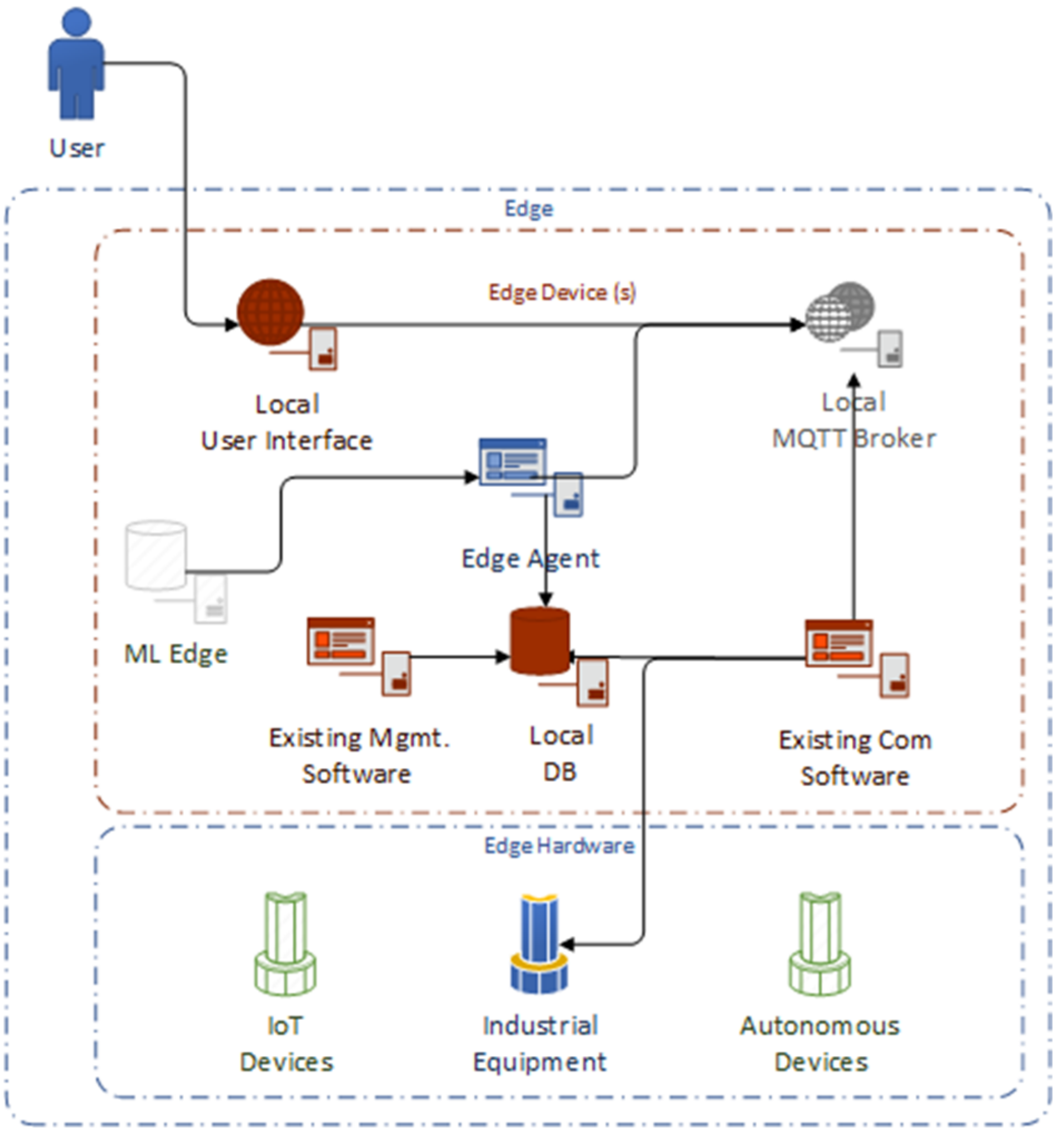


Figure 2. Edge solution architecture.

The architecture depicted in Figure 2 showcases a comprehensive edge computing ecosystem. Here, "Local User Interface" and "Local MQTT Broker" represent the user-interaction and messaging layers, respectively. The "Edge Agent" remains pivotal, interfacing with "ML Edge"—the machine learning component tailored for edge computing. "Existing Management Software" and "Local DB" denote the software for managing operations and the local database for data storage. The inclusion of "Existing Com Software" ensures compatibility and communication with legacy systems.

The workflow delineated in Figures 3 and 4 represents a streamlined solution devised to manage database updates efficiently. This solution's premise is to intercept CRUD (Create, Read, Update, Delete) operations at the API level and channel them into a control database, which then acts as an intermediary in the synchronization process.

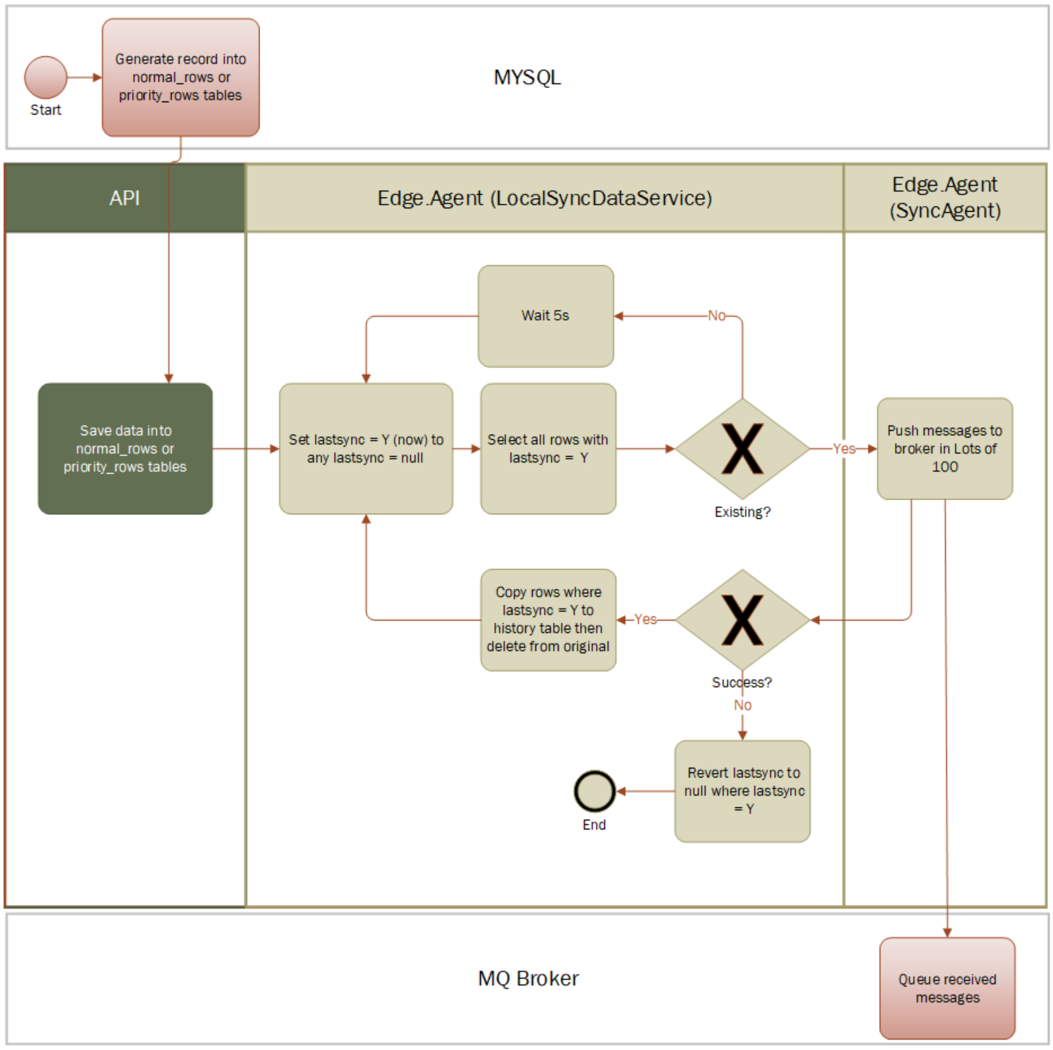

Start
Generate record into normal_rows or priority_rows tables
MYSQL
API
Edge.Agent (LocalSyncDataService)
Edge.Agent (SyncAgent)
Wait 5s
No
Save data into normal_rows or priority_rows tables
Set lastsync = Y (now) to any lastsync = null
Select all rows with lastsync = Y
Existing?
Yes
Push messages to broker in Lots of 100
Copy rows where lastsync = Y to history table then delete from original
Yes
Success?
No
Revert lastsync to null where lastsync = Y
End
MQ Broker
Queue received messages

Figure 3. Cloud architecture mixing new and existing applications to sync MySQL data.

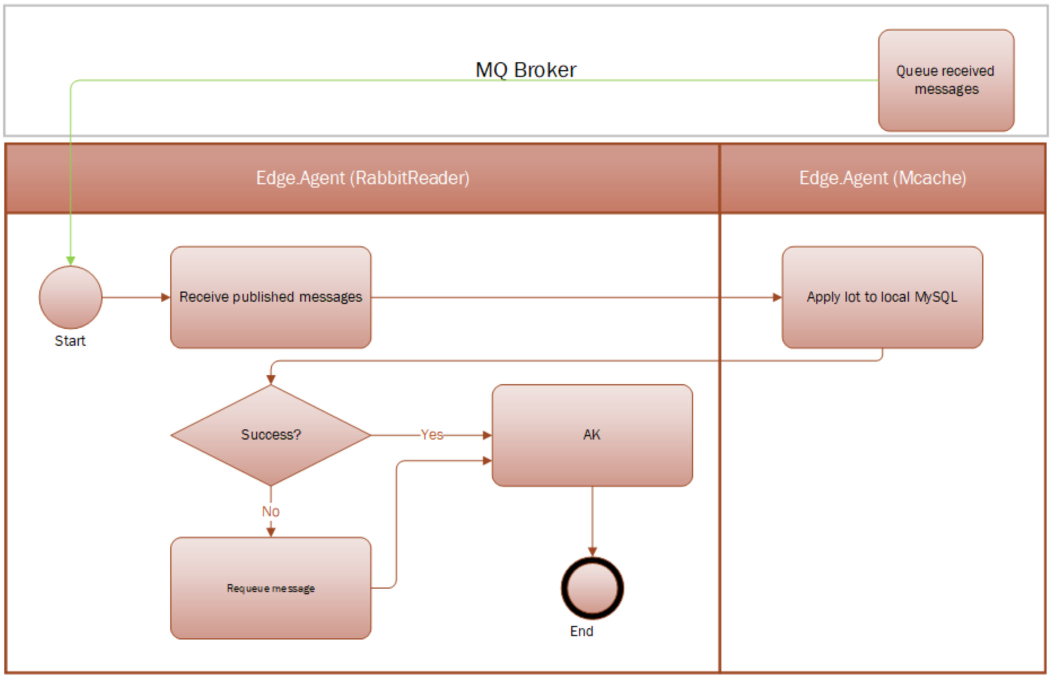


Figure 4. Edge architecture mixing new and existing applications to sync MySQL data.

The implemented mechanism involves extending the current transactional programs to trigger notifications to a dedicated API whenever a record is created, updated, or deleted. These events are captured in real-time and logged into the control database. Here, each record is augmented with "last updated" and "last sync" timestamps, which are critical in tracking changes and determining the data's currentness. A synchronization program operates continuously in the background, polling the control database to ascertain new changes that are ready for transmission. When a change is detected, the system leverages RabbitMQ to queue the data for further operations, ensuring that the updates are processed in an orderly fashion without overloading the system.The queued data are then handled accordingly—either flagged for a refresh from the database or pushed through the local cache for immediate use. This approach is advantageous in scenarios where a push-based update is required, such as in real-time monitoring or when triggering remote commands. Furthermore, the diagram illustrates the use of a memory cache for data retrieval efficiency, reducing redundant reads from the database and enhancing the performance of data-intensive operations. These cached data can be utilized to generate or update multiple CSV files for tasks that require batch processing or offline analysis.

3. Advancing Edge Architecture: Integration and Synchronization Enhancements for Comprehensive Testing

In contrast to the previous iteration, the architecture showcased in Figure 5 represents a pivotal shift towards a MySQL-driven system. This alignment with our industrial partner's infrastructure facilitates a nuanced data synchronization method. Unlike the manual and local storage-based approach of the past, the current system automates the flow of data. It rigorously channels updates from the cloud to the edge and back, ensuring data integrity and consistency across platforms. This method not only captures complete updates but is also adept at handling incremental changes, marking a substantial enhancement in the system's capabilities. Figure 6 details the sequence of actions performed from start to end, moving any updated or inserted data from cloud down to the edge device and once applied into the opposite way but using the same solution it would also perform the edge-to-cloud synchronization.

Subsequent testing of the updated architecture revealed its limitations, particularly in managing partial updates—incremental changes vital for real-time data accuracy. This realization prompted a transition to a more sophisticated system, adept at handling these nuanced updates. The forthcoming section will delve into the intricacies of this refined approach, dissecting how it accommodates the new requirements and the broader implications for our edge computing framework. In the next pages, we explore the depth of this functionality and its integral role in the architecture's evolution. We engaged closely with our industrial partner, whose global reach and deployment of over 150,000 devices across 800 farms in 40+ countries demanded an architecture of unparalleled robustness and scalability. The insights gleaned from these preliminary experiments, conducted over an 18-month period, have been instrumental in refining the design of our edge platform. We explored various hardware configurations, from the modest Raspberry Pi B+ to the powerful Nvidia suite, to identify the ideal characteristics for an edge device capable of handling data synchronization and advanced computational tasks like machine learning.

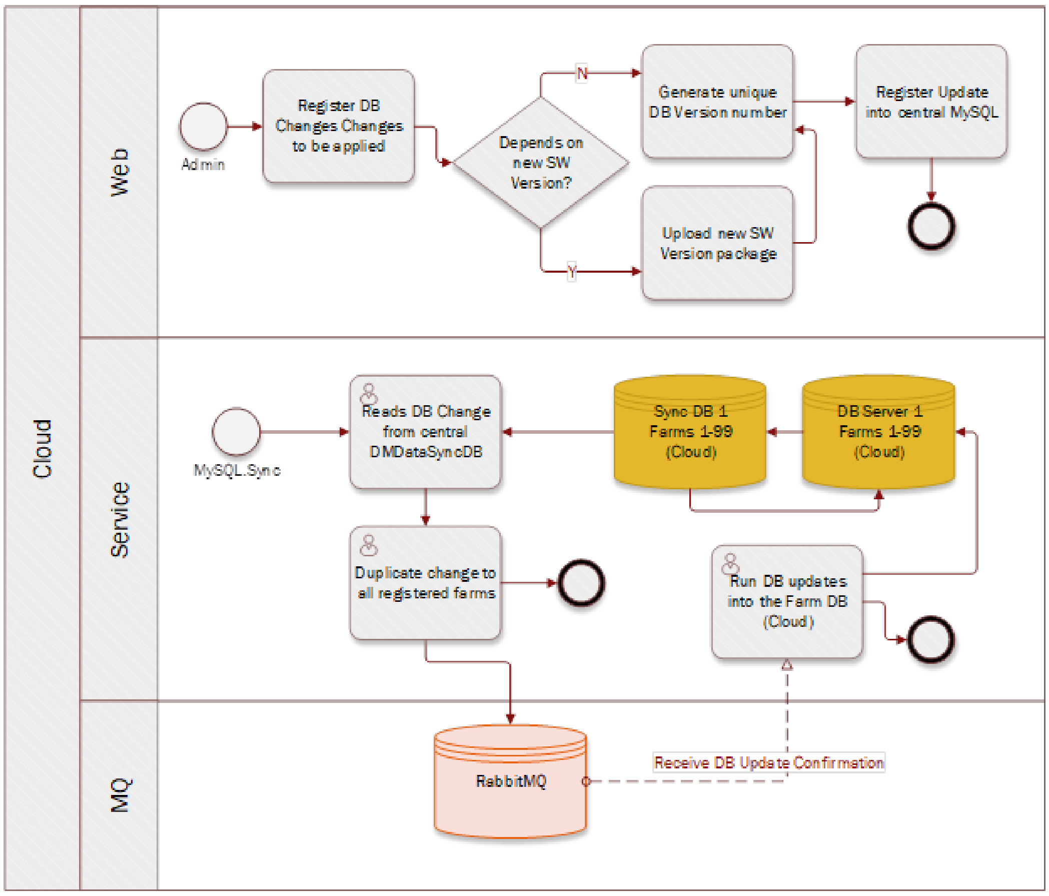

Cloud
Web
Service
MQ
Admin
Register DB Changes Changes to be applied
Depends on new SW Version?
N
Y
Generate unique DB Version number
Upload new SW Version package
Register Update into central MySQL
MySQL.Sync
Reads DB Change from central DMDataSyncDB
Sync DB 1 Farms 1-99 (Cloud)
DB Server 1 Farms 1-99 (Cloud)
Duplicate change to all registered farms
Run DB updates into the Farm DB (Cloud)
RabbitMQ
Receive DB Update Confirmation

Figure 5. Improved cloud architecture.

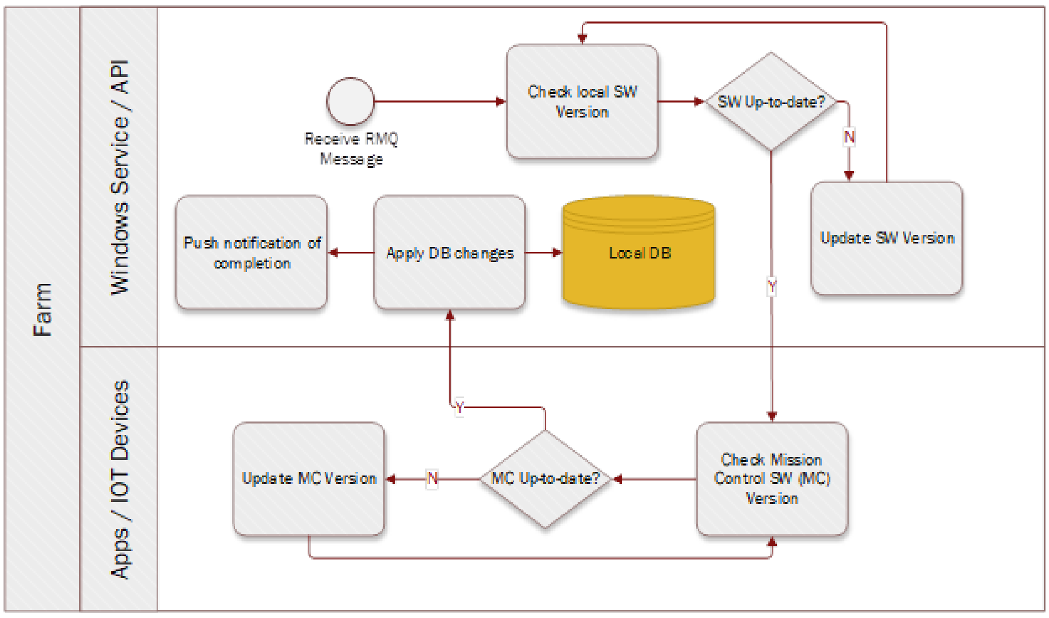
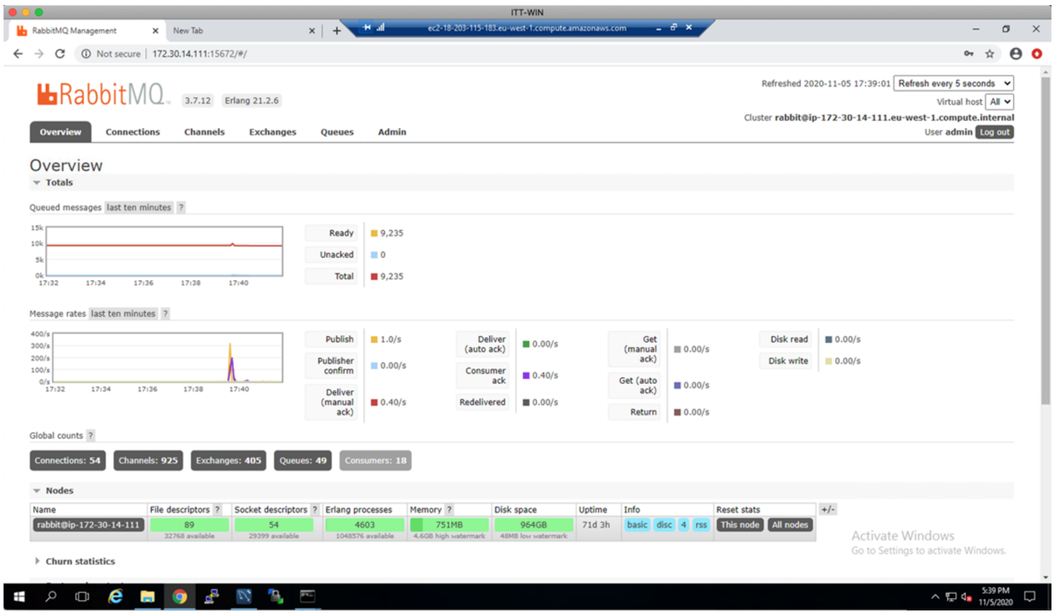


Figure 6. Improved edge architecture.

4. Implementation and Evaluation of the Newly Defined Software Architecture

This section delves into the practical application and empirical assessment of the innovative software architecture introduced in the earlier section. Our focus is on examining how this architecture facilitates efficient data transport between a large-scale cloud database and multiple edge devices, particularly in agricultural settings. By deploying the new solution in both real farms and simulated environments, we aim to offer a comprehensive evaluation of its effectiveness and adaptability. The data captured in Figure 7 mark a pivotal moment in our research, where we touch upon the novel concept at the heart of our project for the first time. It was in fact implemented before we could figure out how novel it was. The problem it was tracking was that did not matter how hard it was tried, the solution would always lose some records due to bad internet connection or software failure at one of the nodes. This innovative mechanism, initiated by setting the "modified_by" field to "ALL", activates the generation of insert commands for the entire dataset within an edge device or at the cloud database, depending on where it is triggered. This initiative is key to enabling a synchronization of data, ensuring that all alterations made at the edge, or to the cloud, are accurately reflected at the other side. This mechanism is more than just a technical routine; it embodies the ingenuity that will be thoroughly dissected later. The methodology paves the way for a new paradigm in cloud–edge database interaction, aligning with our objective to streamline data consistency across distributed environments.



3.7.12 Erlang 21.2.6

Refreshed 2020-11-05 17:39:01 Refresh every 5 seconds
Virtual host All
Cluster **rabbit@ip-172-30-14-111.eu-west-1.compute.internal**
User **admin**

Overview

Totals

Queued messages last ten minutes ?

Ready	9,235
Unacked	0
Total	9,235

Message rates last ten minutes ?

Publish	1.0/s	Deliver (auto ack)	0.00/s	Get (manual ack)	0.00/s	Disk read	0.00/s
Publisher confirm	0.00/s	Consumer ack	0.40/s	Get (auto ack)	0.00/s	Disk write	0.00/s
Deliver (manual ack)	0.40/s	Redelivered	0.00/s	Return	0.00/s		

Global counts ?

Connections: 54 Channels: 925 Exchanges: 405 Queues: 49 Consumers: 18

Nodes

Name	File descriptors ?	Socket descriptors ?	Erlang processes	Memory ?	Disk space	Uptime	Info	Reset stats	+/-
rabbit@ip-172-30-14-111	89 32768 available	54 29399 available	4603 1048576 available	751MB 4.6GB high watermark	964GB 48MB low watermark	71d 3h	basic disc 4 rss	This node All nodes	

Figure 7. MQTT Broker displaying data received from edge device for farms 10001 to 10007.

As shown in Figure 7, all the data were transferred from the edge device to the cloud broker, allowing them to be used at the other end, in this case the cloud. Our research into data synchronization across multiple farming operations has taken a comprehensive approach to test the scalability and performance of the solution. We began by setting up a virtualized environment using Oracle VirtualBox on a Windows Server 2012 machine, which allowed us to emulate the diverse and distributed nature of real-world farm operations. The initial phase involved simulating 11 farms, setting a baseline for our scalability metrics.

To test the robustness of our server—which boasts an I7 12-core processor and 32 GB of RAM—we incrementally increased the number of farms from 11 to 25 and then to 30. This step was critical in understanding how the system managed larger network loads and in pinpointing the limits of our current setup. Through these trials, we determined that the optimal operational threshold for our server configuration was 25 virtual farms, which provided a balance between efficiency and responsiveness. To closely mimic peak operational times such as milking sessions, we conducted stress tests by initiating simultaneous milking processes across all farms. This setup pushed our system to its limits, with CPU utilization nearing 92% and memory usage reaching 21.4 GB, reflecting the system's response to this increased activity. These stress tests were vital in assessing the system's capacity to manage concurrent high-demand tasks.

The role of the MQTT broker was instrumental during these high-activity periods. It efficiently managed a significant surge in data transmission, maintaining an optimized 100:1 ratio of database records to message queuing. This efficiency in handling the influx of data during peak times is illustrated in Figure 8, which shows the results obtained during our experiments, with an average time of 3 s to synchronize data from one end to the other.

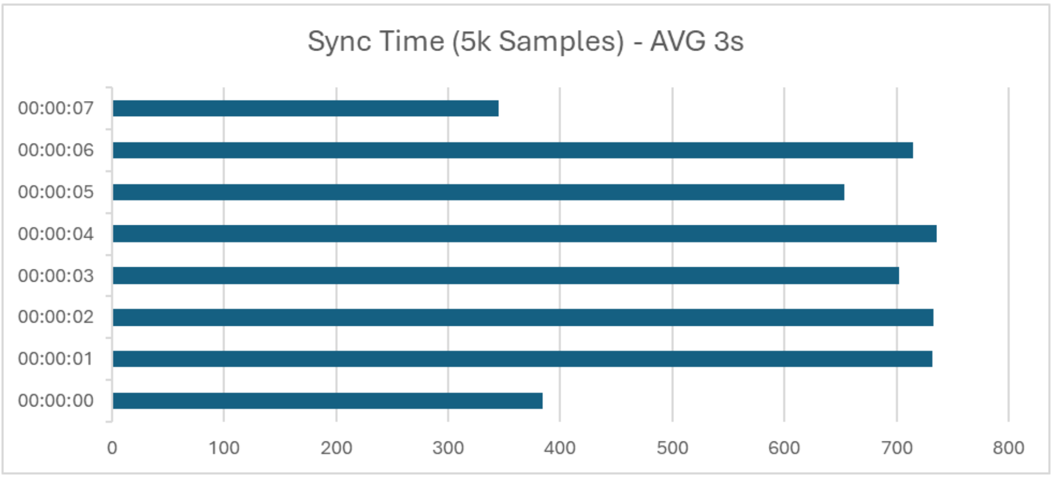
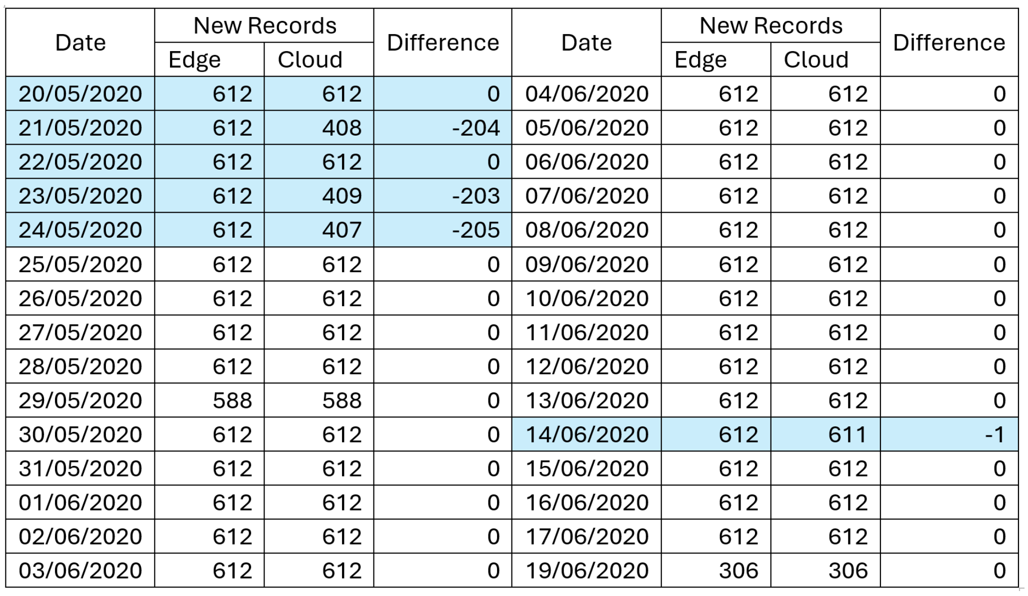


Figure 8. Scalability tests—time to send 5 thousand records from one side to another.

Additionally, we observed a pronounced spike in utilization within the RDS database concurrent with the peak times. This highlighted the successful execution of our data consolidation strategy, which effectively managed the high volumes of data, ensuring they translated into manageable loads on our cloud infrastructure. The information in Figure 9 showcases the synchronization outcomes for Farm 10001 over a 30-day duration, offering an essential insight into the functionality of our synchronization system. During this period, it became evident that while the system generally maintained a consistent match between new records created at the edge and those recorded in the cloud databases, significant data loss occurred during transmission. Particularly noteworthy were the instances on May 24th and 23rd, where the cloud database failed to receive 205 and 203 records, respectively, indicating a notable shortfall in data synchronization. This issue underscores the critical issue of data loss in the transport process from the cloud to the farm.



Date	New Records		Difference	Date	New Records		Difference
	Edge	Cloud			Edge	Cloud	
20/05/2020	612	612	0	04/06/2020	612	612	0
21/05/2020	612	408	-204	05/06/2020	612	612	0
22/05/2020	612	612	0	06/06/2020	612	612	0
23/05/2020	612	409	-203	07/06/2020	612	612	0
24/05/2020	612	407	-205	08/06/2020	612	612	0
25/05/2020	612	612	0	09/06/2020	612	612	0
26/05/2020	612	612	0	10/06/2020	612	612	0
27/05/2020	612	612	0	11/06/2020	612	612	0
28/05/2020	612	612	0	12/06/2020	612	612	0
29/05/2020	588	588	0	13/06/2020	612	612	0
30/05/2020	612	612	0	14/06/2020	612	611	-1
31/05/2020	612	612	0	15/06/2020	612	612	0
01/06/2020	612	612	0	16/06/2020	612	612	0
02/06/2020	612	612	0	17/06/2020	612	612	0
03/06/2020	612	612	0	19/06/2020	306	306	0

Figure 9. Recording sync for 30 days between edge and cloud (Farm 10001).

Through several iterations, our project's coding quality reached a high standard, yet we encountered significant reliability challenges in data transfer. The twin problems were the imperfect transmission from edge devices to the MQTT gateway and flaws in the routines writing data into the AWS Aurora cloud database. AWS Aurora's divergence from MySQL 8, especially in its distributed systems architecture for cloud scalability and reliability, contributed to these challenges. Recognizing these issues was pivotal, leading to targeted enhancements in our synchronization solution. Before commencing further trials, we conducted rigorous refinements to strengthen the system against data loss and ensure it could meet the demands of live deployment scenarios. This process underscored the necessity for a novel approach to address the persistent issue of missing records. These challenges led us to a decisive turning point in our project, necessitating a transformative solution for the persistent issue of missing records. As we transition to the next section, we introduce a new approach to address these synchronization shortcomings.

5. The "PAIR Mechanism"—Ensuring Data Integrity in Edge–Cloud Synchronization

In the intricate sphere of edge computing, particularly within the data-intensive operations of agricultural environments, the synchronization of data between edge devices and cloud databases is a critical aspect of maintaining operational continuity and integrity. The PAIR Mechanism, introduced in this discourse, emerges as a conceptual breakthrough in the field of data consistency and resilience. It is predicated on the principle that systems are inherently fallible, and rather than aspiring for a faultless ecosystem, it seeks to establish robust continuity even amidst failures.

This mechanism is not an isolated development but rather a response to the limitations of existing protocols. While the stalwart TCP/IP suite has provided a foundation for network communication, managing each data packet's integrity becomes increasingly complex in the distributed and varied environments of edge–cloud systems. This complexity is echoed in the works of scholars such as [28], who advocate for architectures that dynamically adapt to network disconnections, and [29], who propose scalable blockchain models to enhance security and efficiency in IoT applications. The PAIR Mechanism aligns with these concepts, focusing on a higher-level goal of achieving consistent data states rather than managing the minutiae of transmission.

Contrary to traditional error-handling methods that meticulously track and correct every packet, the PAIR Mechanism simplifies the process, repeatedly transmitting data until synchronization is confirmed. This brute-force approach might seem rudimentary compared to the elaborate error-correction algorithms described by [30] or the hash-based integrity verification explored by [31]. However, it offers a direct path to data consistency, which is particularly valuable in systems where the environment is fraught with instability and unpredictability.

The PAIR Mechanism's philosophy of resilience and simplicity also resonates with the Distributed Ledger for IoT Data (DLIT) model proposed by [32]. While DLIT emphasizes a two-layered transaction mechanism for improving synchronization in unstable networks, the PAIR Mechanism simplifies this process, streamlining data synchronization by focusing on end-result data consistency. The potential for integrating DLIT's detailed transaction management with the PAIR Mechanism's straightforward strategy could result in a robust framework for managing data in the challenging conditions typical of edge computing.

Furthermore, the real-time reconfiguration capabilities required in dynamic agricultural environments, as discussed by [33], are akin to the PAIR Mechanism's approach of persistent resending. This ensures data freshness and aligns with the principle of the 'Plan for Failure' philosophy. It is within this philosophical framework that the PAIR Mechanism operates, echoing the robust systems design that anticipates and handles errors to maintain system reliability, a concept that is also central to the design of high-availability systems, fault-tolerant databases, and resilient network infrastructures.

In consideration of these comparative frameworks, the PAIR Mechanism offers a novel perspective on the synchronization challenge. By adopting a less-is-more philosophy, it ensures robustness and reliability in data-intensive environments, where traditional models may not offer the most practical solution. It is this innovative approach that may pave the way for efficient and reliable data synchronization in the ever-evolving landscape of edge computing and IoT. Mapping the architecture of the PAIR Mechanism involves understanding its cyclical process of data verification and synchronization between the cloud and edge devices. The mechanism functions on a fundamental assumption that data discrepancies are not just possibilities, but inevitabilities that must be routinely reconciled to maintain system integrity.

As shown in Figure 10, the core of the PAIR Mechanism's workflow is a scheduled task that initiates at predetermined intervals, say every few minutes. This scheduler triggers the cloud starter, which sends a message to the edge device containing the name of the dataset (or table), the number of rows it should have, and an MD5 hash of the dataset. The MD5 hash serves as a fingerprint of the data, providing a compact representation that can be used to verify integrity without transmitting the entire dataset.

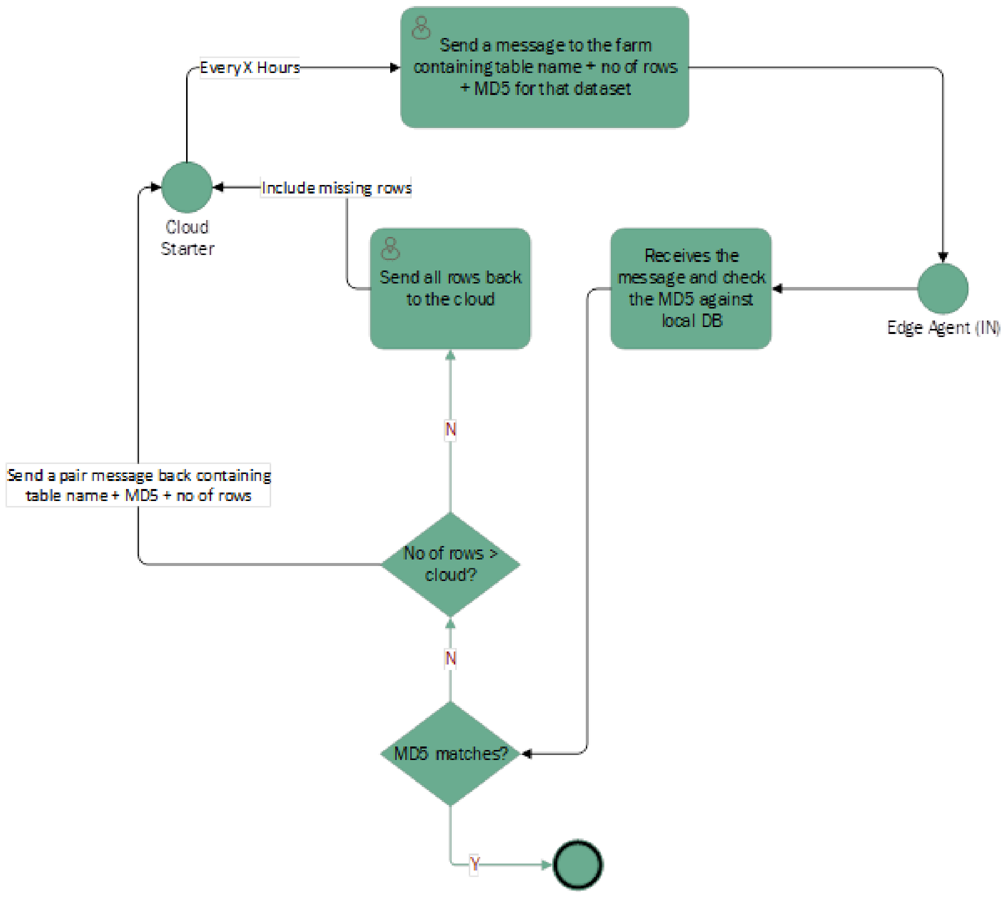

Every X Hours
Send a message to the farm containing table name + no of rows + MD5 for that dataset
Include missing rows
Cloud Starter
Send all rows back to the cloud
Receives the message and check the MD5 against local DB
Edge Agent (IN)
N
Send a pair message back containing table name + MD5 + no of rows
No of rows > cloud?
N
MD5 matches?
Y

Figure 10. The PAIR Mechanism workflow.

Upon receiving this message, the edge agent compares the provided MD5 hash against the hash of the local dataset in its database. If the hashes match, it indicates that the data on the edge device is in sync with the cloud, and no action is needed. This check acts as a lightweight verification step to avoid unnecessary data transmission when both ends are already synchronized. However, if the hashes do not match, the edge agent interprets this as a discrepancy in the datasets. It then enters a reconciliation phase where it sends a message back to the cloud, including the same metadata: the table name, its MD5 hash, and the number of rows from its perspective. The cloud then evaluates this information to determine the next course of action. Should the edge report more rows than the cloud, it suggests that data has been generated at the edge which has not yet been synchronized.

The cloud will then request all rows to be sent back for inclusion in the cloud dataset, thus achieving parity. In cases where the cloud has more rows, it suggests missing data to the edge, and the cloud will send the necessary rows to the edge to fill in the gaps. This back-and-forth communication ensures that both datasets reach a point of equivalence, even if multiple iterations are required. By transferring the entire dataset when discrepancies are detected, the PAIR Mechanism simplifies the process of error checking and data recovery. It is a robust approach, particularly suited for environments where the integrity of the data is crucial, and where the additional bandwidth consumed by transmitting the full dataset is a worthwhile trade-off for the assurance of consistency.

The PAIR Mechanism's approach is markedly different from traditional synchronization methods that may employ more granular data checks and incremental updates. By operating under the assumption that synchronization will fail at some point, it ensures a robust and resilient system that maintains data integrity across distributed nodes. This mechanism is especially relevant in industrial applications where the cost of data inconsistency could be significantly higher than the cost of data transmission, making the PAIR Mechanism a strategic choice for systems where failure is not an option.

How It Improves Synchronization

The PAIR Mechanism improves synchronization through its robust, uncomplicated, and direct approach. Traditional synchronization methods often rely on incremental updates and intricate checks that can become complex and error-prone, especially in distributed systems with a high volume of data transactions. In contrast, PAIR simplifies the process by operating under a bulk transfer strategy that can be more reliable in certain contexts. One of the most significant improvements PAIR offers is its ability to ensure complete data consistency. By resending the entire dataset when discrepancies are detected, PAIR guarantees that both the cloud and edge databases are identical down to the last byte. This is particularly useful in environments where even minor data inconsistencies can lead to significant problems, such as in financial systems or healthcare records management.

Moreover, the PAIR Mechanism's scheduled synchronization checks serve as a proactive measure to maintain data integrity. Instead of waiting for a user or system to flag a data mismatch, which could potentially lead to further complications, PAIR routinely verifies data integrity. This scheduled approach means that issues can be identified and resolved swiftly, minimizing the window in which data inconsistencies can affect system operations. Another advantage of PAIR is its resilience to catastrophic events. In scenarios where an edge device is destroyed or a cloud database is compromised, the bulk transfer method allows for a rapid restoration of the lost or corrupted data. Unlike incremental sync methods that may struggle to reconcile complex historical changes, PAIR's strategy of re-establishing the last known good state is straightforward and effective. The PAIR Mechanism also improves synchronization by reducing the complexity of error handling. In traditional systems, a significant amount of logic is dedicated to managing the various states and scenarios of data mismatches. PAIR's approach eliminates many of these complexities by reducing the number of states the system must handle. This simplification can lead to increased reliability and easier troubleshooting when issues arise.

Finally, PAIR's methodology aligns with the current trends towards more data-driven decision-making processes. As the volume and velocity of data grows, the ability to synchronize large datasets quickly and reliably becomes increasingly important. PAIR's bulk transfer method, despite potentially requiring more bandwidth, ensures that data-driven systems have the up-to-date information they need to function correctly.

6. Conclusions

Our investigation into the realm of agricultural data synchronization has highlighted significant limitations in the applicability of existing methods, such as the Distributed Ledger for IoT Data (DLIT) articulated by [32] within this specific context. These established techniques, while pioneering in their own domains, do not fully address the nuanced challenges encountered in agricultural settings. The introduction of BlockDeepEdge by Rathore et al. [16], the decentralized edge computing framework CoopEdge by Yuan et al. [20], the high-reliability mobile terminal architecture by Ju et al. [21], and the innovative data placement and retrieval service by Xie et al. [22] each contribute to solving aspects of these challenges through distributed deep learning, blockchain technology, and enhanced data management. However, the PAIR Mechanism emerges as a tailored solution to the unique demands of agricultural data management, offering robust synchronization between edge and cloud databases that these other approaches indirectly support but do not specifically address for agricultural applications.

Departing from more complex synchronization frameworks, the PAIR Mechanism provides a stark contrast to the localized beacon synchronization scheme proposed by [34], which, while efficient within its scope, does not cater to the broad-scale synchronization required in the agricultural space. The PAIR Mechanism's methodology eschews the granular control of data packets in favor of robust and repetitive data transmission, ensuring an unerring mirror of data states between edge and cloud databases. This streamlined process, which may appear rudimentary when compared to [34]'s nuanced time-slot allocation for beacon frames, embodies a deliberate move towards simplicity that enhances reliability and operational efficiency in distributed systems.

In our comprehensive review of recent literature, including significant contributions from [16,20–22], we have encountered innovative approaches to edge computing and data synchronization. Despite this, our findings underscore a notable absence of methodologies comparable in simplicity and efficacy to the PAIR Mechanism over the five years since this project's inception. No other proposed solution approaches the unique challenges of agricultural data management with a method as straightforward and effective as PAIR, distinguishing it as a pioneering approach in the field.

The PAIR Mechanism, by simplifying synchronization strategies, marks a significant departure from previously complex frameworks and sets a new benchmark for reliability and efficiency in agricultural data flows. Its success prompts us to consider broader applications beyond agriculture, signaling potential expansions into various sectors. Future research will explore adapting the PAIR Mechanism to different industrial contexts and integrating alternative database technologies, including NoSQL, to enhance versatility and applicability. This direction aims to leverage PAIR's foundational principles to address the dynamic and diverse data synchronization needs across sectors, setting the stage for its evolution as a universal solution for data management challenges.

Author Contributions: Conceptualization, J.W.; Methodology, A.C., D.R. and J.W.; Software, A.C.; Validation, A.C. and D.R.; Formal analysis, A.C.; Investigation, A.C.; Resources, D.R.; Data curation, A.C.; Writing—original draft, A.C.; Writing—review & editing, D.R.; Supervision, D.R. and J.W.; Project administration, D.R. and J.W.; Funding acquisition, J.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Science Foundation Ireland grant 13/RC/2094_P2.

Data Availability Statement: Data are contained within the article.

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

References

1. Pal, A.; Purushothaman, B. *IoT: Technical Challenges and Solutions*; Artech House: London, UK , 2017; pp. 64–65.
2. Taneja, M.; Byabazaire, J.; Davy, A.; Olariu, C. Fog assisted application support for animal behavior analysis and health monitoring in dairy farming. In Proceedings of the 2018 IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 5–8 February 2018; pp. 819–824.
3. Bittencourt, L.F.; Lopes, M.M.; Petri, I.; Rana, O.F. Towards Virtual Machine Migration in Fog Computing. In Proceedings of the 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), Krakow, Poland, 4–6 November 2015; pp. 1–8.

4. Varshney, P.; Simmhan, Y. Demystifying Fog Computing: Characterizing Architectures, Applications and Abstractions. In Proceedings of the 2017 IEEE 1st International Conference on Fog and Edge Computing (ICFEC), Madrid, Spain, 14–15 May 2017; pp. 115–124.
5. Rimal, B.P.; Choi, E.; Lumb, I. A Taxonomy and Survey of Cloud Computing Systems. In Proceedings of the 2009 Fifth International Joint Conference on INC, IMS and IDC, Seoul, Republic of Korea, 25–27 August 2009; pp. 44–51.
6. Huo, R.; Yu, F.R.; Huang, T.; Xie, R.; Liu, J.; Leung, V.C.M.; Liu, Y. Software Defined Networking Caching and Computing for Green Wireless Networks. *Commun. Mag. IEEE* **2016**, *54*, 185–193. [CrossRef]
7. Caprolu, M.; Di Pietro, R.; Lombardi, F.; Raponi, S. Edge Computing Perspectives: Architectures, Technologies, and Open Security Issues. In Proceedings of the 2019 IEEE International Conference on Edge Computing (EDGE), Milan, Italy, 8–13 July 2019; pp. 116–123. [CrossRef]
8. Botta, A.; de Donato, W.; Persico, V.; Pescap, A. Integration of cloud computing and internet of things: A survey. *Future Gener. Comput. Syst.* **2016**, *56*, 684–700. [CrossRef]
9. Giang, N.K.; Lea, R.; Blackstock, M.; Leung, V.C.M. Fog at the Edge: Experiences Building an Edge Computing Platform. In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 9–16.
10. Wolfert, J.; Sørensen, C.G.; Goense, D. A Future Internet Collaboration Platform for Safe and Healthy Food from Farm to Fork. In Proceedings of the 2014 Annual SRII Global Conference, San Jose, CA, USA, 23–25 April 2014; pp. 266–273.
11. Ashjaei, M.; Bengtsson, M. Enhancing smart maintenance management using fog computing technology. In Proceedings of the 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), Singapore, 10–13 December 2017.
12. Wan, J.; Chen, B.; Wang, S.; Xia, M.; Li, D.; Liu, C. Fog Computing for Energy-Aware Load Balancing and Scheduling in Smart Factory. *IEEE Trans. Ind. Inform.* **2018**, *14*, 4548. [CrossRef]
13. Sharma, R.; Biookaghazadeh, S.; Li, B.; Zhao, M. Are Existing Knowledge Transfer Techniques Effective for Deep Learning with Edge Devices? In Proceedings of the 2018 IEEE International Conference on Edge Computing (EDGE), San Francisco, CA, USA, 2–7 July 2018; pp. 42–49.
14. Thakkar, K.; Patel, A.; Patel, S.; Patel, K.; Nayak, A.; Budhrani, A. A Complete Virtual Edge Computing Extrapolation: Architectural Style, Uses, and Implications. In Proceedings of the 2023 IEEE World Conference on Applied Intelligence and Computing (AIC), Sonbhadra, India, 29–30 July 2023. [CrossRef]
15. Chiaraviglio, L.; D'Andreagiovanni, F.; Lancellotti, R.; Shojafar, M.; Melazzi, B.; Canali, N.; An C. Approach to Balance Maintenance Costs and Electricity Consumption in Cloud Data Centers. *IEEE Trans. Sustain. Comput.* **2018**, *3*, 274–288. [CrossRef]
16. Rathore, S.; Sharma, P.K.; Rathore, H. A Distributed Deep Learning Approach with Mobile Edge Computing for Next Generation IoT Networks Security. In Proceedings of the 2023 World Conference on Communication & Computing (WCONF), Raipur, India, 14–16 July 2023. [CrossRef]
17. Busse, M.; Schwerdtner, W.; Siebert, R.; Doernberg, A.; Kuntosch, A.; König, B.; Bokelmann, W. Analysis of animal monitoring technologies in Germany from an innovation system perspective. *Agric. Syst.* **2015**, *138*, 55–65. [CrossRef]
18. Sjaak, W.; Lan, L.; Cor, V.; Marc-Jeroen, B. Big Data in Smart Farming—A review. *Agric. Syst.* **2017**, *153*, 69–80.
19. Caria, M.; Schudrowitz, J.; Jukan, A.; Kemper, N. Smart farm computing systems for animal welfare monitoring. In Proceedings of the 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, Croatia, 22–26 May 2017; pp. 152–157.
20. Yuan, L.; He, Q.; Tan, S.; Li, B.; Yu, J.; Chen, F.; Yang, Y. CoopEdge: Enabling Decentralized Secure and Cooperative Multi-Access Edge Computing Based on Blockchain. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 894–908. [CrossRef]
21. Ju, B.; Liu, Y.; Song, L.; Gan, G.; Li, Z.; Jiang, L. A High-Reliability Edge-Side Mobile Terminal Shared Computing Architecture Based on Task Triple-Stage Full-Cycle Monitoring. *IEEE Internet Things J.* **2023**, *10*, 20149–20161. [CrossRef]
22. Xie, J.; Qian, C.; Guo, D.; Li, X.; Wang, G.; Chen, H. A Novel Data Placement and Retrieval Service for Cooperative Edge Clouds. *IEEE Trans. Cloud Comput.* **2023**, *11*, 71–84. [CrossRef]
23. Sahadevan, A.; Mathew, D.; Mookathana, J.; Jose, B.A. An Offline Online Strategy for IoT Using MQTT. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 369–373.
24. Hunkeler, U.; Truong, H.L.; Stanford-Clark, A. MQTT-S—A publish/subscribe protocol for Wireless Sensor Networks. In Proceedings of the 2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08), Bangalore, India, 6–10 January 2008; pp. 791–798.
25. De Caro, N.; Colitti, W.; Steenhaut, K.; Mangino, G.; Reali, G. Comparison of two lightweight protocols for smartphone-based sensing. In Proceedings of the 2013 IEEE 20th Symposium on Communications and Vehicular Technology in the Benelux (SCVT), Namur, Belgium, 21 November 2013; pp. 1–6.
26. Buyya, R.; Khan Pathan, A.-M.; Broberg, J.; Tari, Z. A Case for Peering of Content Delivery Networks. *IEEE Distrib. Syst. Online* **2006**, *7*, 3. [CrossRef]
27. Grozev, N.; Buyya, R. Performance modelling and simulation of three-tier applications in Cloud and Multi-Cloud environments. *Comput. J.* **2015**, *58*, 1–22. [CrossRef]

28. Le, M.; Song, Z.; Kwon, Y.-W.; Tilevich, E. Reliable and efficient mobile edge computing in highly dynamic and volatile environments. In Proceedings of the 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), Valencia, 8–11 May 2017; pp. 113–120. [CrossRef]
29. Bazari, A.S.; Singh, A.; Khan, A.A.; Jindal, R. Filter Based Scalable Blockchain for Domestic Internet of Things. In Proceedings of the 2020 5th International Conference on Communication and Electronics Systems (ICCES), Coimbatore, India, 10–12 June 2020; pp. 1051–1056. [CrossRef]
30. Ghimire, S.; Choi, J.Y.; Lee, B. Using Blockchain for Improved Video Integrity Verification. *IEEE Trans. Multimed.* **2020**, *22*, 108–121. [CrossRef]
31. Hegde, N.; Manvi, S. Hash Based Integrity Verification for Vehicular Cloud Environment. In Proceedings of the 2019 8th IEEE International Conference on Cloud Computing in Emerging Markets, CCEM 2019, Bengaluru, India, 19–20 September 2019; pp. 75–79. [CrossRef]
32. Niya, S.R.; Beckmann, R.; Stiller, B. DLIT: A Scalable Distributed Ledger for IoT Data. In Proceedings of the 2020 Second International Conference on Blockchain Computing and Applications (BCCA), Antalya, Turkey, 2–5 November 2020; pp. 100–107. [CrossRef]
33. Misic, J.; Misic, V.B.; Banaie, F. Reliable and Scalable Data Acquisition from IoT Domains. In Proceedings of the GLOBECOM 2017—2017 IEEE Global Communications Conference, Singapore, 4–8 December 2017; pp. 1–6. [CrossRef]
34. Choudhury, N.; Matam, R.; Mukherjee, M.; Lloret, J. LBS: A Beacon Synchronization Scheme with Higher Schedulability for IEEE 802.15.4 Cluster-Tree-Based IoT Applications. *IEEE Internet Things J.* **2019**, *6*, 8883–8896. [CrossRef]