Journal of
Sensor and Actuator Networks

Article

Mobile IoT-Edge-Cloud Continuum Based and DevOps Enabled Software Framework

Janis Judvaitis *,†, Rihards Balass † and Modris Greitans †

Institute of Electronics and Computer Science, 14 Dzerbenes St., LV-1006 Riga, Latvia;
rihards.balass@edi.lv (R.B.); modris_greitans@edi.lv (M.G.)
* Correspondence: janis.judvaitis@edi.lv
† These authors contributed equally to this work.

Abstract: This research aims to provide a high-level software framework for IoT-Edge-Cloud computational continuum-based applications with support for mobile IoT and DevOps integration utilizing the Edge computing paradigms. This is achieved by dividing the system in a modular fashion and providing a loosely coupled service and module descriptions for usage in the respective system layers for flexible and yet trustworthy implementation. The article describes the software architecture for a DevOps-enabled Edge computing solution in the IoT-Edge-Cloud computational continuum with the support for flexible and mobile IoT solutions. The proposed framework is validated on an intelligent transport system use case in the rolling stock domain and showcases the improvements gained by using the proposed IoT-Edge-Cloud continuum framework.

Keywords: IoT; Edge; Edge computing; DevOps; mobile IoT; rolling stock

check for updates

Citation: Judvaitis, J.; Balass, R.; Greitans, M. Mobile IoT-Edge-Cloud Continuum Based and DevOps Enabled Software Framework. *J. Sens. Actuator Netw.* **2021**, *10*, 62. https://doi.org/10.3390/jsan10040062

Academic Editor: Mohammad Hammoudeh

Received: 2 August 2021
Accepted: 27 October 2021
Published: 30 October 2021

1. Introduction

In the past decade, the concept of IoT has emerged and grown quite substantially, and it is predicted that it will continue to grow. For example, Grand View Research predicts that the Narrow Band IoT(NB-IoT) market size will reach more than $6 billion by 2025 [1]. This is the trend we are monitoring in our everyday life around us, e.g., everything is becoming digitalized and smart, IoT and big data trends are growing fast [2]. However, regardless of the target application of the specific IoT deployment, the security and trustworthiness of the solution must not be undermined. As a recent study on IoT security [3] observes, there are countless attack vectors on IoT systems, which can lead not only to malfunction of the system but has the possibility of being weaponized [4,5]. Not surprisingly, the same issues persist in other parts of the system necessary for the IoT system to function properly, including data delivery, processing and storing. To overcome this, a new paradigm called IoT-Cloud continuum [6] has emerged in recent years.

In this article, we are describing our findings and research on a mobility-enabled IoT-Edge-Cloud continuum in the direction of Intelligent Transport Systems, with a practical focus on the rolling stock digitalization process using the IoT-Edge-Cloud continuum approach, and we are limiting the scope mostly to the IoT and Edge part of the system. The existing literature in the IoT-Edge-Cloud continuum field mostly focuses on computation offloading solutions [7–9], and in this article, we add the mobility of IoT/Edge nodes and integration of DevOps approaches to the IoT-Edge-Cloud computational continuum with a real use case.

This work tries to address the following problems: (i) create a vertically integrated approach to the IoT-Edge-Cloud continuum development pushing for faster and simpler development phases, (ii) enabling sustained operation of the system even in the case of loss in cloud connectivity allowing for more robust and flexible deployments and (iii) provide an easier way of conceptualization and implementation of mobility-enabled high processing power capable devices promoting trustworthiness of the solution by bringing the data processing closer to the data source.

As a challenging use case we decided on an application in the intelligent transport system domain, specifically the rolling stock monitoring application. Rolling stock is a perspective and growing possibility of transportation. The International Transport Forum on their ITF Transport Outlook 2019 [10] predict that rail passenger transport will grow faster than all other transportation means, achieving an annual growth rate of 3.7%, and by 2050 it will generate approximately 47 trillion passenger-kilometers, outpacing all other means of transportation. In 2017, the overview of East Japan Railway Company's development [11] was published stating that they are striving towards an intelligent rolling stock utilizing IoT and AI technologies benefiting the passengers with safety, comfortability and peace of mind. The developed solution was partly used for the validation of DevOps enablers described in the [12] as part of the described ITS use case.

The rest of the document is structured as follows: Section 3 describes the proposed IoT-Edge-Cloud continuum framework in general, Section 4 describes the implementation on the ITS domain, and finally, Section 5 provides a discussion about the proposed framework and its future applicability.

2. Related Work

The increase of IoT devices has highlighted one of the most concerning problems in mobility-enabled IoT devices - their lack of computational and battery power. There has been an introduction of several new methods of computing, such as Mobile Edge computing [13] and Fog Computing [14]. One strategy to improve IoT systems is to integrate IoT devices with edge and cloud in a single architecture End-Edge-Cloud [15]. The workload and memory management in IoT systems is the key to their efficiency and is achieved by introducing small operating systems [16]. An introduction to edge computing itself has given the opportunity of taking the major load off of IoT devices, mainly so that it would save power [6]. Edge devices are located in between the IoT and cloud, so it is necessary to elevate the edge layer to be at the same level as the cloud in terms of communication [17], so that edge devices can communicate with each other, thus removing hard cloud dependency from the equation.

A popular method of work distribution is osmotic computing [18] where computational load is taken off of the cloud services and moved to edge devices. The latest development in this regard is the IoTSim-Osmosis, which simulates IoT networks [19] to help distribute the computing load between the edge devices. Another approach to IoT systems is the integration of Artificial Intelligence. This is possible because the edge devices have become more powerful and can execute more complex and computationally heavy tasks [20].

The work described in [21] describes an architecture pattern for trusted orchestration management in an edge cloud, which somewhat relates to the ideas expressed in this article, but the scope is limited to the edge cloud and not the whole IoT-Edge-Cloud continuum. It is noteworthy that there is interesting evidence that trust and security concerns can be mapped using a blockchain-based solution at an architecture level. To the best of our knowledge there is no similar IoT-Edge-Cloud continuum software framework described in the literature.

3. Framework Architecture

To tackle the business logic of managing maintenance, communications, logistics and safety of the mobility-enabled IoT solution, we are proposing an IoT-Edge-Cloud continuum DevOps-enabled software framework. This means that the framework should operate on all levels of the computational continuum, make use of available resources smartly and confine within the DevOps practices.

When describing the proposed software framework, we chose to document data flow paths and services operating in the continuum, such representation allows abandoning the concept of a single device and look at the greater picture, giving a better understanding of the mechanisms in action. The system is separated in the three main layers: (i) IoT, (ii) Edge and (iii) Cloud. In the edge and cloud layer, we separate two modules: (i) operational logic module, referred to as **OL Edge** and **OL Cloud** and (ii) business logic module, referred as **BL Edge** and **BL Cloud**. The operational logic module is responsible for managing the data flows and processing the technical information of the system, while the business logic module is purely for the external integration with the system. In the command and data flow path, the OL Cloud is always located between the BL Cloud and edge layer and the OL Edge is located between the BL Edge and IoT layer, as shown in Figure 1.

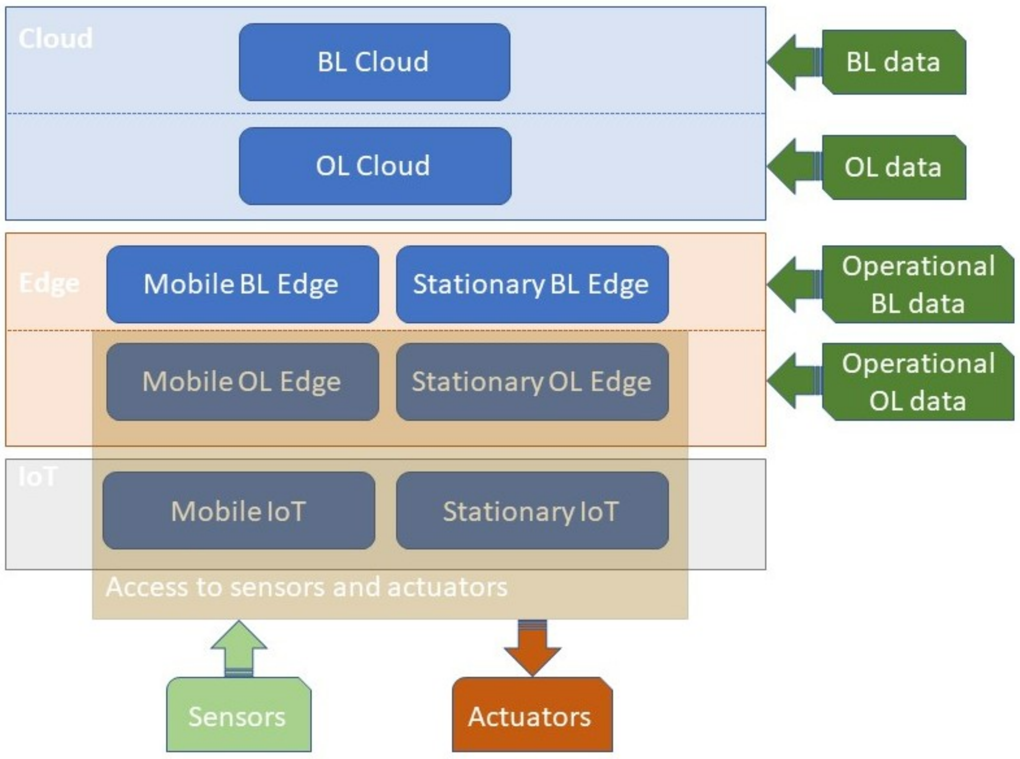

Figure 1. Proposed mobility-enabled IoT-Edge-Cloud continuum data flow diagram.

The relationship between the same modules located on the cloud and edge layer is complementary, the modules on the cloud layer have virtually unlimited computation and power resources and thus they are responsible for the heavy lifting. However, the corresponding modules on the edge layer are more constrained with processing power and often have additional power constraints, but they are located closer to the data sources and thus can react to changes much faster. The framework is planned in such a way that once the data reaches the edge layer, which is physically located on the premises, the initial processing and decision making is performed then and there. If any actuation or maintenance actions, such as notifying the personnel or activating a device, is necessary, the edge layer can make that decision and provide a real-time response to the IoT layer about changes. In parallel, the raw data received from the IoT layer are also forwarded to the cloud layer for a thorough inspection and some additional processing that is not feasible for the edge layer. Once the edge layer has processed the data and provided the inputs to the IoT layer, it also informs the cloud layer about the actions taken, this allows the cloud layer to oversee the process and intervene if the limited decision making on the edge layer has reacted in a sub-optimal way. This intervention can come in the form of direct intervention into the system operation as soon as possible if it is necessary or as a directive for the edge layer to alternate the processing algorithm to avoid such sub-optimal behavior in the future.

To support the mobility factor in our proposed IoT-Edge-Cloud continuum framework, we need to draw the line between stationary and mobile layers. Typically it is assumed the IoT layer can be mobile while the edge layer is not suited for mobility, but we try to move the edge layer into the mobile part of the system. This introduces a set of challenges and provides benefits to the solution. The challenges that are the most obvious—power and connectivity—need to be sorted at the infrastructure level, but the solution should be constraint-aware and act accordingly. The main benefit is the possibility to move processing closer to the mobile solution, which is closer to the source of data, thus reducing the latency, improving privacy and most importantly dependency on external connectivity. This is especially helpful in the scenarios where the IoT devices are located on a moving object, for example, car or pedestrian, and have the possibility to warn the user or intervene but lack the processing power to compute the necessary action. Introducing a mobile edge layer, capable of providing the additional processing power to the premises of data source, enables smart system functionality without the latency and availability issues of cloud-only based solutions.

3.1. Services

The framework we propose consists of multiple interconnected software services, each providing a distinct functionality to the smart IoT system. Each service hosts modules necessary for the operation of the system at the selected level. Note that service may be operational in one or more layers, but the module is limited to a specific layer. We categorize the software modules by the corresponding layer of where they are located on the IoT-Edge-Cloud continuum and each of them has a specific task, which promotes modularity in the software framework and the developed system in general.

We propose a software framework with six distinct services: (i) data service, (ii) actuation service, (iii) processing service, (iv) logistics service, (v) maintenance service and (vi) status service. The overview of services and their possible corresponding layers of operation can be seen in Figure 2.

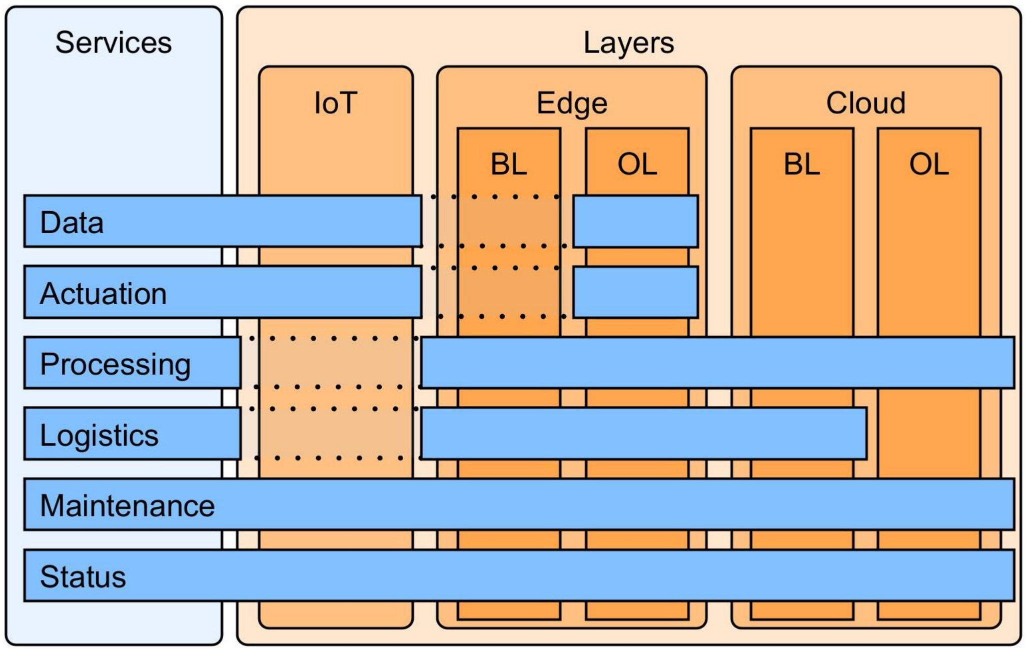

Figure 2. Possible service distribution across different system layers.

Data service hosts modules operating on IoT and OL Edge layers, both stationary and mobile. The tasks of the modules hosted on the data service are related to data acquisition from sensors connected to the IoT devices and possibly also edge devices. The additional metadata, such as timestamps, etc., complementing the gathered data are also generated on this service.

Actuation service hosts modules operating on IoT and OL Edge layers, both stationary and mobile, similarly to the data service, but the tasks for hosted modules concern the actuation of system components. This service should include some sort of global actuation conflict manager, which is easy to implement since the service distinguishes the actuation subsystem from the rest of the system.

Processing service hosts modules on stationary and mobile edge and cloud layers. The hosted modules are the main data processing points in the IoT-Edge-Cloud continuum, promoting the distribution of computation between the edge and cloud layers. The processing modules can be self-contained within the layer and also share the computational load with modules from other layers, as necessary.

Logistics service hosts modules operating on stationary and mobile edge layers and the BL Cloud layer. The modules hosted on this service provide the logistics information about the monitored environment. This information can be used by the BL modules for task specific operations.

Maintenance service hosts modules on all the layers included in the system, providing deployment and feedback capabilities thus allowing for the DevOps practices to be used seamlessly with the developed IoT-Edge-Cloud continuum system.

Status service hosts modules operating on all layers included in the system. The modules hosted on the status service provide real-time status information about the deployed IoT and edge infrastructure, signaling any services or modules not working properly and faults raised in the system. We have not included the operation of this service on the cloud layer because of the complexity of load managing and redundancy involved, since it is out of the scope of this work.

3.2. DevOps Integration

The proposed mobility-enabled IoT-Edge-Cloud continuum software framework does not provide a way of using DevOps practices from day one, instead the path to DevOps integration is shown via the characteristics and usage of maintenance and status services. Thus, the responsibility for initial steps ensuring the DevOps practices falls on the developers of the system, but once the crucial modules are implemented, the proposed framework encourages to use the feedback received from the system through the maintenance and status modules and update the system using the development module.

The traditional DevOps approach consists of seven stages, as shown in Figure 3. The status service corresponds with the monitor stage of the DevOps framework providing the developer and operation manager with the complete information about the system, allowing to detect any problems or inconsistencies faster. The maintenance service operates on the release and deploy and operate stages of the DevOps framework providing the necessary support at the phase where the system needs to be updated and configured. The overall approach of the proposed IoT-Edge-Cloud continuum software framework workflow has a high correlation with the stages of DevOps by design, so the two would naturally blend together.

3.3. Advantages

There are two types of advantages of using the proposed mobility-enabled IoT-Edge-Cloud continuum software framework related to:

- Continuous operation during connectivity operations.
- Ability to move the data processing closer to the data source by using the edge layer mobility.

The framework facilitates the ability to continue to function in the case of loss of cloud connectivity, which is a crucial trait for a system being deployed in any safety critical application, for example, intelligent transport system, e-health, etc. The proposed framework achieves this by moving the crucial part of processing away from the cloud and into the edge layer. When this is achieved, the deployed system can continue to function based only on the processing performed locally, enabling the most important processes to continue while leaving the non-critical ones to wait for the return of cloud connectivity, whereas the traditional approach is not capable of continuing to operate in the case of connectivity issues.

Another novelty is introducing a mobile edge layer, which removes the limitation of static location from the otherwise mobile and, more often than not, a battery-powered IoT layer, while still providing the aforementioned ability to operate in the case of connectivity loss. The mobile and possibly battery-powered part of the system has a much higher risk of losing the connectivity due to the non-robust nature of wireless networks over large distances. This is the application scenario where the proposed framework shines the most, because the system can be mobile and operational irrespective of external factors.

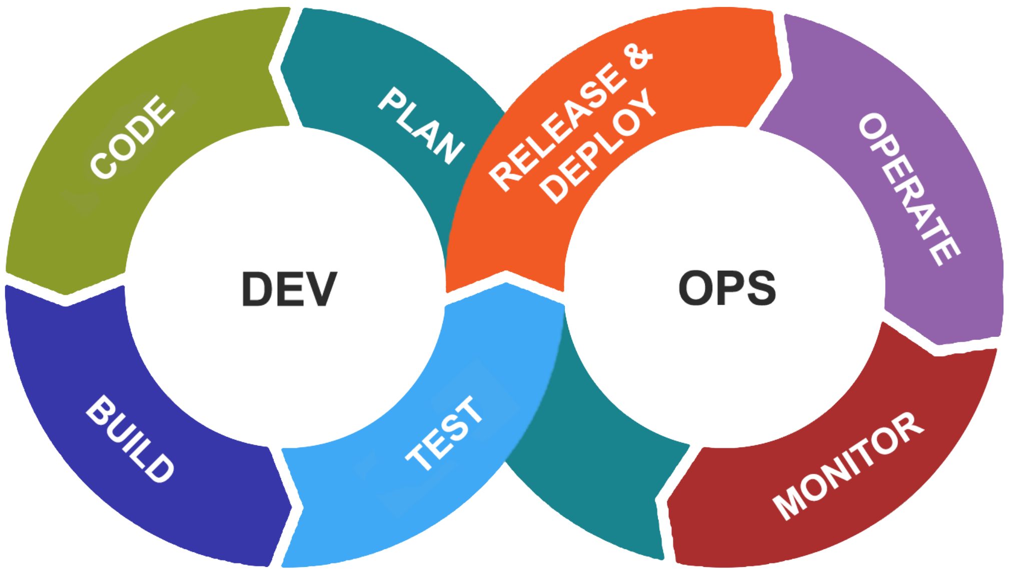

Figure 3. DevOps diagram.

4. Evaluation and Methodology

The used methodology to evaluate and test the proposed solution was designed around the practical implementation and evaluation of the minimum required IoT-Edge-Cloud continuum infrastructure based on a use case in the Intelligent Transport System domain. For the validation, we use the fact that the implemented system is operational and that the data received across the layers corresponds to the initial plans and the system is able to execute the necessary actuation commands. The use case is showcasing the usage of our proposed IoT-Edge-Cloud continuum framework in the smart rolling stock monitoring system, which allows to demonstrate the flexibility and strengths of our solution. The use case provides rolling stock inauguration procedure, rolling stock integrity control, logistics and maintenance data for business logic and DevOps practices showcase the usability of the proposed IoT-Edge-Cloud framework. The system was tested in a lab environment and on a test-bench capable of providing data accordingly to TRL5. In this section, we first introduce the reasoning behind the developed use case and then provide the implementation details. The workflow of the implementation is summarized in Figure 4.

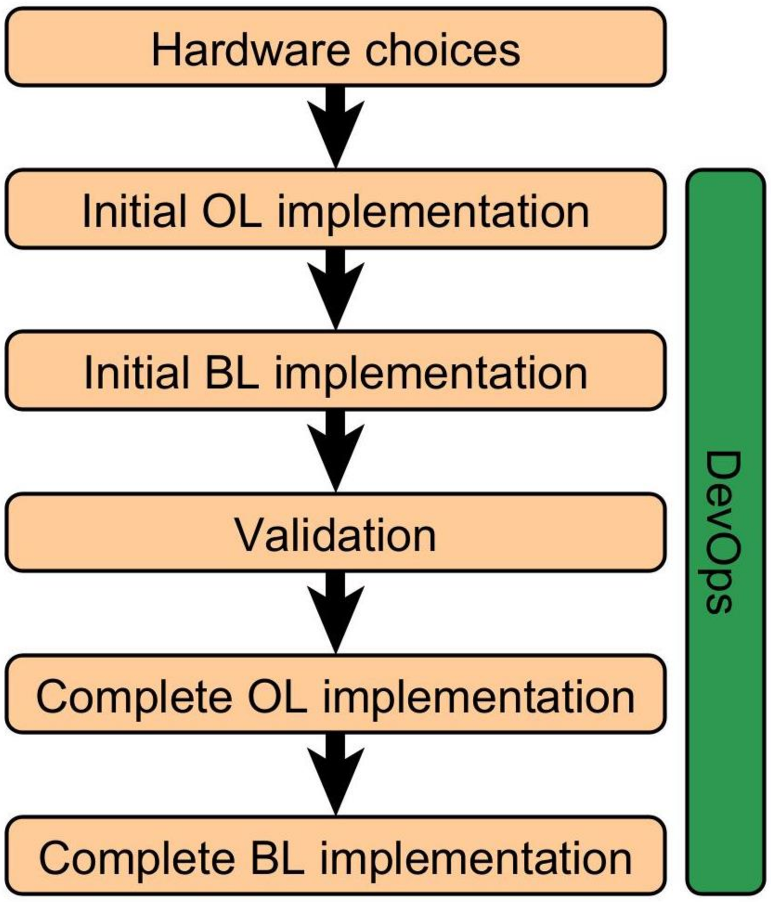
Hardware choices
Initial OL implementation
Initial BL implementation
Validation
Complete OL implementation
Complete BL implementation
DevOps

Figure 4. Mobility-enabled IoT-Edge-Cloud continuum software architecture workflow.

4.1. Mobility-Enabled IoT and Edge Devices

When installing IoT devices called sensor nodes on rolling stock, the device count should match the wagon count, because in an ideal solution, we would like to monitor all of the wagons. Since usually the wagons of the rolling stock are located close to each other, it makes no sense to equip each of the sensor nodes with communication capabilities outside of the rolling stock proximity, it is much more practical to use a single sensor node, called a base station, with the ability to send the data gathered from the rolling stock to the next processing point. There are two reasonable possibilities of placing the base station either (i) on the locomotive or (ii) in the middle of the train. The placement of base station on the locomotive has two main advantages: (i) usually the locomotive is the only place on a freight train where there is any sort of electricity connection or at least a train operator next to it, and (ii) the setup on a train is simpler because at the depot it might be non-trivial to find the center wagon of the rolling stock, and trains often change the count of wagons multiple times during a single voyage, but a locomotive usually always travels with the train. Of course the sensor nodes and base station should be placed outside of the wagon for better communication capabilities, which means that the enclosures used should be at least IP67 certified. To sum up, we are developing our Iot-Edge-Cloud continuum software framework demonstrator based on the assumptions that there is a single sensor node located on each of the rolling stock wagons, and the base station is located on the locomotive.

The base station itself should be physically the same as the rest of the sensor nodes to lower the cost and complexity of the solution with the only differences being in the software, which will be described later, and the fact that the base station should be connected to the on-board gateway. With the on-board gateway, we understand the device with the following capabilities: (i) can communicate to the base station, usually through a USB connection, (ii) has a connection to the internet, directly or through any other device or modem (this makes no difference with respect to the evaluated framework), and (iii) has some processing power (typically such devices are Linux powered, like Raspberry Pi, Jetson Nano or similar).

Regarding the sensors and actuators on the rolling stock, we operate with the minimum set of Global Navigation Satellite System (GNSS), Accelerometer and Radio-Frequency Identification (RFID) reader/writer. This set is required to demonstrate the basic functionality of a smart rolling stock monitoring system.

4.2. Stationary IoT and Edge devices

We also use on-track infrastructure for rolling stock monitoring. The minimal implementation we showcase is just for a redundancy purpose on the inauguration and integrity systems already working with the data provided by the on-board infrastructure, but we believe that complementing the on-track infrastructure with additional sensors or actuators could be beneficial in the future, for example, railway track switching, etc. The rolling stock is moving through check points, where the wagon count and order are recorded. For example, when the rolling stock is exiting the depot, passing the on-track infrastructure, the actual order of wagons can be verified.

The hardware used in the on-track infrastructure does not differ much from the on-board one. The IoT and Edge layers are integrated into a single device we call on-track gateway. The on-track gateway can accommodate the RFID reader and since it uses more powerful batteries or, depending on the location, is connected to the electricity grid, which also allows connecting a bigger external antenna for extended range needed to interact with the RFID tags located on the rolling stock and possibly moving with high velocity. There is no need for a base station or sensor nodes on the on-track infrastructure. Typically, the same hardware as in on-board gateway can be used for an on-track gateway. The use of Power over Ethernet (PoE) can possibly bring additional benefits when deploying the on-track infrastructure.

4.3. Cloud Solutions

As for the requirements of cloud infrastructure, this very much depends on the scale of the solution. Processing capabilities needed for the data received are minimal, but the processing latency should also be kept minimal, mostly for safety reasons. This means that any queuing of the processing tasks should be avoided at all costs. As for the storage capacity, there is no overflow of data coming from the rolling stock, and this is more specific to actual business implementation and not that relevant for the smart IoT system itself.

4.4. Hardware

We chose the specific hardware parts based on availability and ease of access, fully knowing that the solution in this iteration would never be used on actual rolling stock in real operational conditions. We do not encourage anyone to use the same hardware and warn against using any hardware that is incompatible with railway standard EN 50155/IEC 60751 [22].

To ease the initial implementation and prototyping phase, we used EDI WSN/IoT TestBed [23] functionality, which allowed us to implement and test the core services before the complete IoT infrastructure was implemented. This testbed also encourages DevOps usage [24], which aligns with the aim of the proposed framework. EDI TestBed uses the MQTT communication protocol and gives access to manage end devices, and we used this to set up a mock-up on-board and on-track infrastructures and forward the data between services with MQTT. More detailed usage will be described in the following sections for each of the layers.

4.4.1. IoT Hardware

The first iteration of the IoT layer implementation was performed by using EDI TestBed default sensor nodes Advanticsys XM1000, which provided dummy data in the correct format for the on-board system. Since the only connection to the edge layer is from the base station, we could have just simulated all of the on-board wireless sensor networks with a single EDI TestBed workstation, where the base station would provide the dummy data for all nodes, but we chose to use one workstation per sensor node to better represent the possible delays and errors introduced by the wireless network used to communicate from sensor nodes to the base station.

For the second iteration, we added the minimum necessary sensors to the solution—accelerometer, GNSS and RFID reader. With the addition of these sensors, the default sensor node provided by the EDI TestBed was not enough, so we switched to Nucleo Board and added the XBEE shield for communication capabilities. This solution sent data over the air using the IEEE 802.15.4-based ZigBee protocol for secure and reliable communications over 868 MHz medium. The devices were encapsulated in an IP67 rated housing with external antenna. As for the RFID reader, there are multiple methods for asset tracking, such as bar-codes, QR-codes, RFID tagging, Bluetooth or GNSS tracking, etc. All of these methods are semi or fully automated, such as for bar-codes and QR-codes, and there is still a need for a person who scans the codes. However, for Bluetooth and GNSS, everything is automated. GNSS does not work indoors, such as a depot or warehouse, and the precision is not ideal for asset tracking. A Bluetooth beacon system for asset tracking could work, but for it to work, we would need to create an additional node, with a Bluetooth module and a power source for every asset we have to track. RFID options look promising, for they only have to be placed on the asset and then read by any compatible reader. Passive tags require no power source and are usually low cost. There are multiple RFID frequency options, all with their own reading distance, as shown in Table 1.

Table 1. RFID frequency options.

Name	Frequency	Distance	Notes
Low Frequency	120–150 KHz	10 cm	
High Frequency	13.56 MHz	10 cm–1 m	
Ultra High Frequency	433 MHz	1–100 m	Active tags
Ultra High Frequency	865–868 MHz	1–12 m	Passive tags, Europe
Microwave	2.45–5.8 GHZ	1–2 m	Active tags
Microwave	3.1–10 GHZ	up to 200 m	Passive tags

For our tests, we decided to go with the UHF passive RFID system, because this kind of system is already being used for railroad tracking [25].

The third iteration was intended to simultaneously showcase the usage of heterogeneous sensor nodes and multiple actual data sources for the proposed IoT-Edge-Cloud continuum framework. As the sensor nodes, we chose the NRF52840 board and added only the Siemens BMO055 accelerometer. The rest of the data were simulated by the sensor nodes, as they are not reliable while the sensors are not located on a real rolling stock.

4.4.2. Edge Hardware

For the first iteration, we used the Linux-based Alix routers included in the EDI TestBed workstations. Since they already forwarded all of the data from connected sensor nodes to the MQTT broker, minimal changes were necessary for the proof-of-concept prototype. We did not use an on-track gateway in the first iteration.

For the second iteration, we moved away from the EDI TestBed, so we needed a new on-board gateway hardware, and thus, we chose to use Raspberry Pi 4 since it was fully compatible with the software developed for the EDI TestBed workstation. Raspberry Pi 4 also has the pin-out to support the RFID reader we used for our on-board sensor nodes, so we could use the same type of hardware for the on-track gateway. We powered the on-board gateway with batteries and on-track gateway through the PoE.

4.4.3. Cloud Solution

We used a virtual machine running Linux to implement the services running on the cloud layer, for the first iteration it was connected to the EDI TestBed MQTT broker, and with the second iteration, we hosted a local MQTT broker on this machine.

4.5. Validation Test-Bench

We used a mini rolling stock test-bench shown on Figure 5 to validate the feasibility of our proposed mobility-enabled IoT-Edge-Cloud continuum-based software framework for rolling stock monitoring applications. The demo setup was used to provide the sensor, logistics and maintenance data from the on-board and on-track infrastructure while located in the lab environment but still providing non-generated data, thus making it easier to test, integrate, debug, and showcase developed IoT-Edge-Cloud continuum framework on a rolling stock demo.

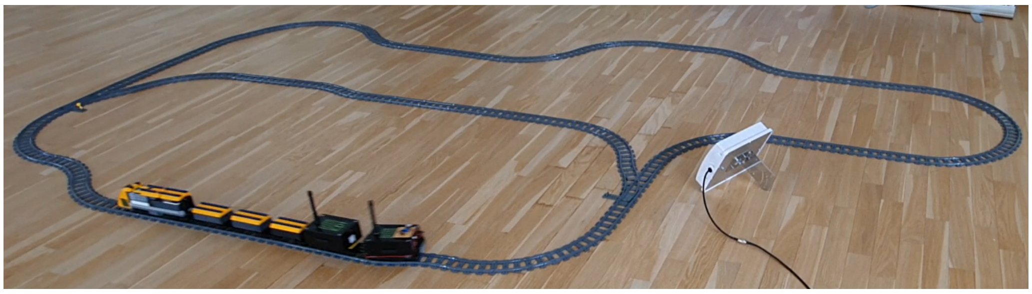

Figure 5. Mini rolling stock test-bench.

4.6. Software

The framework demonstrator consists of multiple interconnected software modules, each providing a distinct functionality to the smart IoT system for rolling stock application. We categorize the software modules by the corresponding layer of where they are located on the IoT-Edge-Cloud continuum. The complete deployed module scheme can be seen In Figure 6.

The software IoT layer consists of two distinct software modules: **sensor node** and **base station**. As per a typical wireless sensor network, the sensor nodes gather the data and send it to the coordinator, i.e., the base station, which forwards the data to the edge layer. The edge layer is responsible for data processing and forwarding from the actual data source to the storage. We chose to use MQTT broker Mosquitto, because this gives us an easy-to-set-up solution, with flexibility for future changes if there is a need.

4.6.1. Data Service

This service hosts an on-board sensor module in the mobile IoT layer and an on-track sensor module on the stationary edge layer.

On-board sensor module: The data are gathered at the sensor nodes located on each wagon of the rolling stock. Data are sent over the air using a reliable bi-directional communications channel. All the data transmissions are encrypted and use a Cyclic Redundancy Check (CRC). Above that, no other means of data validation, storing or processing are present on the IoT level.

On-track sensor module: The on-track data gathering does not require the wireless sensor and actuator network, only the RFID sensor readings are gathered in this module, and writing of RFID tags is not allowed for on-track infrastructure. The reason is that RFID tags should be treated as read-only while the rolling stock is not located in the depot.

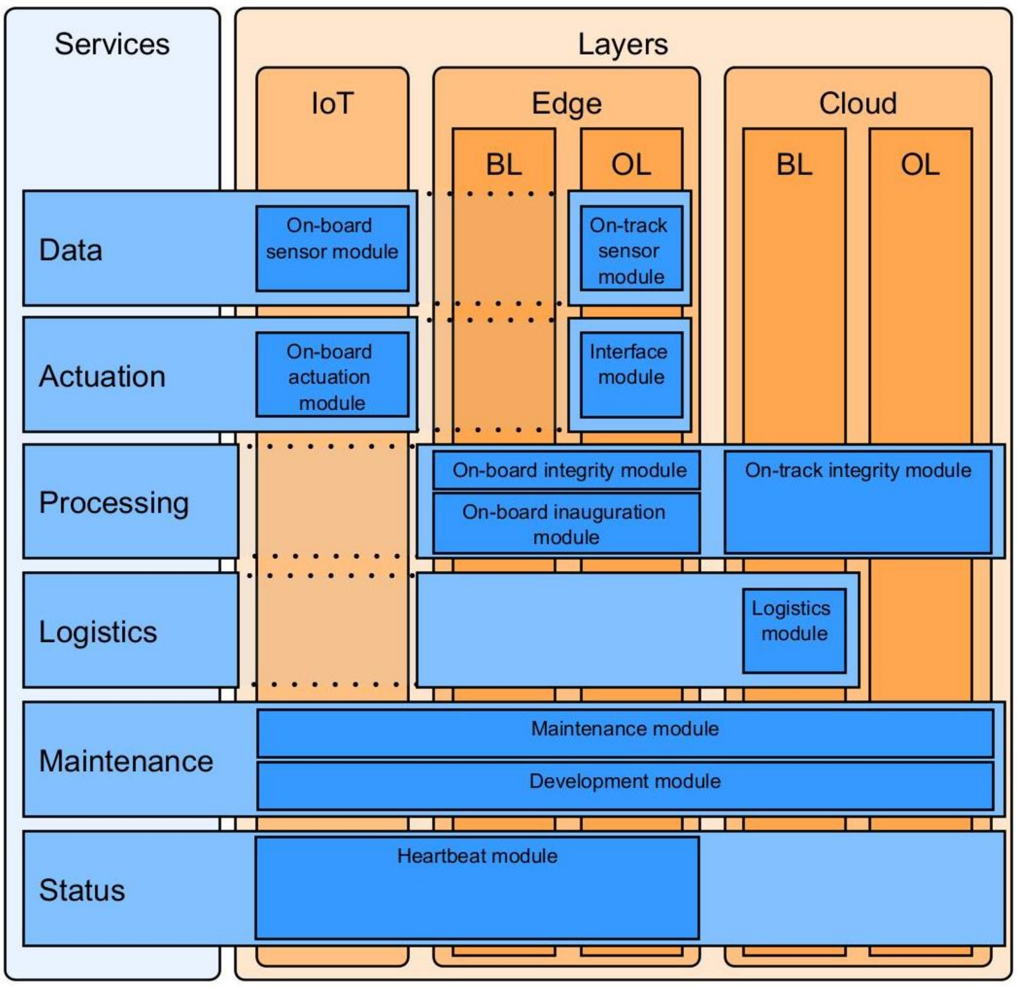

Figure 6. Module deployment scheme.

4.6.2. Actuation Service

This service hosts the on-board actuation module in the mobile IoT layer and interface module on the mobile edge layer.

On-board actuation module: This module supports actuation commands for the (i) RFID tag writing originating from the logistics service, (ii) the inauguration procedure starting from the processing service and (iii) the integrity control starting from the processing service.

Interface module: This module is intended for the rolling stock operator interface, displaying information and alerts about the rolling stock. This module receives information from processing, logistics, maintenance and status services and displays relevant information to the rolling stock operator.

4.6.3. Processing Service

This service hosts an on-board integrity module and on-board inauguration module in the mobile edge layer and on-track integrity module on the stationary edge layer.

On-board integrity module: This module is responsible for continuously validating the rolling stock integrity while it is operational. Integrity control can only be started after a successful inauguration procedure is finished and integrity control is not already operational. When the on-board integrity module receives the command to start the integrity control, it notifies the on-board actuation module to send out the integrity start command. After the start command, each of the sensor nodes belonging to the current rolling stock and base station reports the sensor status through the on-board sensor module every 250 ms or 4 times per second. The received sensor data consist of the GNSS position, accelerometer data and RSSI value measured at the base station for each wagon. Using this information, the train integrity is calculated at the on-board integrity module at the mobile edge layer, which is based on the aforementioned sensors, and train integrity is considered lost in the case when at least two of three sensors report data that indicate that there is a train integrity issue. The train integrity information is communicated to the logistics, maintenance and status services. In the case of any issues, the on-board integrity module immediately reports the issue to the rolling stock personnel using the Interface module located on the mobile edge layer. The train integrity system remains operational until the stop or reset command is received. If the start inauguration or start integrity command is received by the on-board integrity module while the integrity system is operational, the command is ignored.

On-board inauguration module: The on-board inauguration module is implemented on the mobile edge layer. After receiving the command about the start of the inauguration procedure, the on-board actuation module is notified to identify the sensor nodes located on the wagons belonging to this rolling stock and validate that they are operational. As the response to the inauguration command, the on-board sensor module sends the information about sensor nodes found in the proximity. If no response is received after 15 s, the inauguration procedure is assumed to be failed. Once the information is received on the on-board inauguration module, the physical order of the wagons is calculated using the GNSS and RSSI sensor values and RFID data. After successful inauguration, the inauguration status is reported to the logistics, maintenance and status services with high-level information about the composition of rolling stock.

On-track integrity module: This module receives data from the on-track sensor module and validates that the wagon order corresponds to the order reported when the rolling stock performed the inauguration procedure at the depot. The validation result is forwarded to the logistics, maintenance and status services and displayed to the rolling stock operator through the interface module.

4.6.4. Logistics Service

This service hosts the logistics module in the cloud layer.

Logistics module: The logistics module is responsible for providing the business information and control over the rolling stock logistics. The logistics module relies on the RFID reader data provided by the on-board and on-track sensor modules and RFID tag information writing functionality provided by the on-board actuation module located. The on-track sensor module provides wagon order information when the rolling stock is passing the infrastructure located on the track-side. The pn-board sensor module provides information about the cargo and also allows to identify the wagon by the RFID tag associated with it. The logistics module can be started after successful inauguration and stopped at will.

4.6.5. Maintenance Service

This service hosts maintenance and development modules on each of the system layers. Both modules are intended for external intervention in the system by maintenance staff or developers.

Maintenance module: The maintenance module is responsible for providing the maintenance information about the rolling stock and its monitoring infrastructure. The maintenance module relies on the information provided by the on-board infrastructure. In the minimal setup, while the maintenance module is operational, it reports the sensor node energy consumption and battery charge status, but can be extended to report data from various sensors located on the rolling stock. The maintenance module can be started after the successful inauguration procedure and stopped at will. Furthermore, the maintenance module handles the reset command and will reset the IoT and/or Edge layer software in case of any operational faults detected. The reset command will not be accepted if the inauguration process is running or the integrity control module is operational.

Development module: The development module allows for updating of the software running in the corresponding layer in a fail-safe manner, ensuring that the update is successful or performing a rollback in a case of failure. The development module can only trigger an update process if the rolling stock is located at the depot and the system is not performing any actions.

4.6.6. Status Service

This service hosts the heartbeat module on mobile IoT and mobile and stationary edge layers.

Heartbeat module: This module periodically checks the status of all modules deployed on the same layer and reports the status to the processing and maintenance Services. This allows to monitor the framework's overall state and trigger an alert if any of the modules are not operating properly.

4.7. Validation

The Intelligent Transport System developed by using our proposed software framework was validated in the different experiments in the scope of the H2020 ENACT project, where the used IoT-Edge-Could software framework provided the flexibility and modularity necessary to make crucial changes to the behavior of the system. One example of the system in action https://www.youtube.com/watch?v=9ABYxu37StA (accessed on 29 October 2021) demonstrates the gathering and processing of data in an unexpected event. The experiment consists of data generated and processed by our IoT and edge layers and the behavioral drift analysis [26] concluded on the cloud layer.

5. Results and Discussion

The provided mobility-enabled IoT-Edge-Cloud continuum and DevOps enabled software framework aims to solve the main challenges when a trustworthy and secure IoT system is implemented with a complete vertical integration in mind. The proposed framework provides a high-level abstraction over the system allowing developers to use a modular approach.

The proposed mobility-enabled IoT-Edge-Cloud continuum software framework was evaluated by implementing a rolling stock monitoring application in an intelligent transport system domain. The six defined services were used to implement 11 modules necessary for the rolling stock monitoring application, utilizing both the stationary and mobile aspects of the framework and showcasing the possibilities of moving the data processing closer to the originating data source. Over the two iterations, the applicability of the DevOps approach was demonstrated by first using EDI TestBed, migrating to standalone hardware, updating the used modules and adding new functionality accordingly. The implementation described in Section 4 validates that the proposed software framework can be used to implement a non-trivial Smart IoT System integrated in the IoT-Edge-Cloud computational continuum with support for mobile IoT devices and DevOps practices.

The implementation showcases that the problems we are trying to solve with the proposed IoT-Edge-Cloud continuum are not trivial and require much planning in order to achieve an optimal result. The use of the proposed framework allowed for earlier implementation of the minimum viable product and combined with the DevOps approach provided faster initial results. Once the operational logic is implemented, the system becomes operational at all layers, and business logic implementation can take place while the system is being operated.

The innovative contributions of the research are: (i) a vertically integrated approach to the architecture of the IoT-Edge-Cloud continuum, (ii) enabling of sustained operation without the cloud connectivity during the development and deployment phases in the whole DevOps life cycle and (iii) addition of the mobility-enabled edge layer for more flexible and robust data processing closer to the source of the data.

DevOps and full integration of the rolling stock monitoring solution into the IoT-Edge-Cloud continuum is necessary for a secure and trustworthy solution. As we have shown, the development of a rolling stock monitoring solution imposes certain challenges, which our proposed secure and trustworthy IoT-Edge-Cloud continuum-based software framework addresses. The distinction between operational logic and business logic on the cloud layer provides the flexibility for the framework to be used in different scenarios, be it freight or passenger trains, regarding the rolling stock applications, or other use cases in intelligent transport systems or other domains. The possibilities of emerging trends of edge computing were investigated in the prospect of increasing the cargo and passenger privacy and safety when using the rolling stock infrastructure. The presented use case showcased the implementation and its advantages, being able to continue the operation despite the loss of cloud connectivity as well as taking advantage of the mobility-enabled edge layer. The aforementioned would be possible by using conventional methods only after the additional work required by the development team to specifically implement these features, whereas the proposed framework already includes the baseline architecture required.

The mobility-enabled edge layer can become a reality in part thanks to the emerging possibilities of energy harvesting as well as the ever-evolving capabilities of batteries. This has been tackled in the Horizon 2020 ENACT project [12,27].

For the future research directions, we see the possibility to load balance the workload between edge and cloud layers and usage of AI-enabled solutions on the edge to provide the deployed system with cloud-like processing capabilities with edge-like latency using the knowledge distillation approach. Furthermore, the possibility of a two-layer system should be explored, where the edge layer is capable of reacting fast with an approximate result and the cloud layer is providing a final, more precise decision with higher latency. The applications for such a solution would be able to start the reaction to triggering events faster than a cloud-only solution but would still benefit from the precise decision of a cloud layer, even in the case of a mistake in the edge layer processing.

Author Contributions: Conceptualization, M.G. and J.J.; methodology, J.J.; software, J.J.; validation, J.J. and R.B.; formal analysis, M.G.; investigation, J.J.; resources, R.B.; writing—original draft preparation, J.J.; writing—review and editing, M.G.; visualization, R.B.; supervision, M.G.; project administration, M.G.; funding acquisition, M.G. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by European Union's H2020 Programme under grant agreement No. 780351.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

IoT	Internet of Things
OL	Operational logic
BL	Business logic
RFID	Radio-frequency identification
RSSI	Received signal strength indicator
GNSS	Global Navigation Satellite System
MQTT	Message Queuing Telemetry Transport
UFH	Ultra high frequency
PoE	Power over Ethernet
CRC	Cyclic Redundancy check

References

1. NB-IoT Market Size Worth $6.02 Billion by 2025. Available online: https://www.bloomberg.com/press-releases/2019-07-23/nb-iot-market-size-worth-6-02-billion-by-2025-cagr-34-9-grand-view-research-inc (accessed on 3 August 2021).
2. Hajjaji, Y.; Boulila, W.; Farah, I.R.; Romdhani, I.; Hussain, A. Big data and IoT-based applications in smart environments: A systematic review. *Comput. Sci. Rev.* **2021**, *39*, 100318. [CrossRef]
3. Hassan, W.H. Current research on Internet of Things (IoT) security: A survey. *Comput. Netw.* **2019**, *148*, 283–294.
4. Timothy, J.; Hahn, M.A. The Weaponization of IoT Devices: Rise of the Thingbots. Available online: https://www.ibm.com/downloads/cas/6MLEALKV (accessed on 3 December 2021).
5. Harssema, E.C. Weaponization of IoT Devices. Ph.D. Thesis, Utica College: Utica, NY, USA, 2020.
6. Bittencourt, L.; Immich, R.; Sakellariou, R.; Fonseca, N.; Madeira, E.; Curado, M.; Villas, L.; DaSilva, L.; Lee, C.; Rana, O. The internet of things, fog and cloud continuum: Integration and challenges. *Internet Things* **2018**, *3*, 134–155. [CrossRef]
7. Hong, Z.; Chen, W.; Huang, H.; Guo, S.; Zheng, Z. Multi-hop cooperative computation offloading for industrial IoT–edge–cloud computing environments. *IEEE Trans. Parallel Distrib. Syst.* **2019**, *30*, 2759–2774. [CrossRef]
8. Long, J.; Luo, Y.; Zhu, X.; Luo, E.; Huang, M. Computation offloading through mobile vehicles in IoT-edge-cloud network. *Eurasip J. Wirel. Commun. Netw.* **2020**, *2020*, 1–21. [CrossRef]
9. Yu, M.; Liu, A.; Xiong, N.N.; Wang, T. An intelligent game based offloading scheme for maximizing benefits of IoT-edge-cloud ecosystems. *IEEE Internet Things J.* **2020**. [CrossRef]
10. Forum, I.T. *How Transport Demand Will Change by 2050*; International Transport Forum: Leipzig, Germany, 2019 [CrossRef]
11. Asano, K. JR east high-speed rolling stock development. *JR East Tech. Rev.* **2017**, *36*, 1–6.
12. Ferry, N.; Song, H.; Metzger, A.; Rios, E. *DevOps for Trustworthy Smart IoT Systems*; Now Publishers: Norwell, MA, USA, 2021 [CrossRef]
13. Abbas, N.; Zhang, Y.; Taherkordi, A.; Skeie, T. Mobile Edge Computing: A Survey. *IEEE Internet Things J.* **2018**, *5*, 450–465. [CrossRef]
14. Yi, S.; Li, C.; Li, Q. A Survey of Fog Computing: Concepts, Applications and Issues. In Proceedings of the 2015 Workshop on Mobile Big Data (Mobidata '15), Hangzhou, China, 22–25 June 2015; Association for Computing Machinery: New York, NY, USA, 2015; pp. 37–42. [CrossRef]
15. Ren, J.; Zhang, D.; He, S.; Zhang, Y.; Li, T. A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet. *ACM Comput. Surv.* **2019**, *52*. [CrossRef]
16. Musaddiq, A.; Zikria, Y.B.; Hahm, O.; Yu, H.; Bashir, A.K.; Kim, S.W. A Survey on Resource Management in IoT Operating Systems. *IEEE Access* **2018**, *6*, 8459–8482. [CrossRef]
17. Ramachandran, U.; Gupta, H.; Hall, A.; Saurez, E.; Xu, Z. Elevating the Edge to Be a Peer of the Cloud. In Proceedings of the 2019 IEEE 12th International Conference on Cloud Computing (CLOUD), Milan, Italy, 13–19 July 2019; pp. 17–24.
18. Villari, M.; Fazio, M.; Dustdar, S.; Rana, O.; Ranjan, R. Osmotic Computing: A New Paradigm for Edge/Cloud Integration. *IEEE Cloud Comput.* **2016**, *3*, 76–83. [CrossRef]
19. Alwasel, K.; Jha, D.N.; Habeeb, F.; Demirbaga, U.; Rana, O.; Baker, T.; Dustdar, S.; Villari, M.; James, P.; Solaiman, E.; et al. IoTSim-Osmosis: A framework for modeling and simulating IoT applications over an edge-cloud continuum. *J. Syst. Archit.* **2021**, *116*, 101956. [CrossRef]
20. Masip-Bruin, X.; Marín-Tordera, E.; Sánchez-López, S.; Garcia, J.; Jukan, A.; Juan Ferrer, A.; Queralt, A.; Salis, A.; Bartoli, A.; Cankar, M.; et al. Managing the Cloud Continuum: Lessons Learnt from a Real Fog-to-Cloud Deployment. *Sensors* **2021**, *21*. [CrossRef] [PubMed]
21. Pahl, C.; El Ioini, N.; Helmer, S.; Lee, B. An architecture pattern for trusted orchestration in IoT edge clouds. In Proceedings of the 2018 Third International Conference on Fog and Mobile Edge Computing (FMEC), Barcelona, Spain, 23–26 April 2018; pp. 63–70.
22. THE RAILWAY STANDARDS DIN EN 50155/IEC 60751. Available online: https://www.wago.com/global/railway-systems/railway-standard (accessed on 31 March 2021).
23. Ruskuls, R.; Lapsa, D.; Selavo, L. Edi wsn testbed: Multifunctional, 3d wireless sensor network testbed. In Proceedings of the 2015 Advances in Wireless and Optical Communications (RTUWO), Riga, Latvia, 5–6 Nov 2015; pp. 50–53.
24. Judvaitis, J.; Nesenbergs, K.; Balass, R.; Greitans, M. Challenges of DevOps Ready IoT Testbed. In Proceedings of the MDE4IoT/ModComp@ MoDELS; CEUR-WS.org, Munich, Germany, 15–17 September 2019; pp. 3–4.
25. RFID and Rail: Advanced Tracking Technology. Available online: https://www.railway-technology.com/features/feature1684/ (accessed on 10 March 2021).
26. Rocher, G.; Lavirotte, S.; Tigli, J.Y.; Cotte, G.; Dechavanne, F. An IOHMM-Based Framework to Investigate Drift in Effectiveness of IoT-Based Systems. *Sensors* **2021**, *21*, 527. [CrossRef] [PubMed]
27. Solberg, A.E.A. Case Studies Implementation—Final Version. 2021. Available online: https://enact-project.eu/deliverables/D1.5.pdf (accessed on 10 September 2021).