An Introduction to Edge Computing and A Real-Time Capable Server Architecture

Volkan Gezer, Jumyung Um, and Martin Ruskowski
Innovative Factory Systems (IFS)
German Research Center for Artificial Intelligence (DFKI)
Kaiserslautern, Germany
Emails: {name.surname}@dfki.de

***Abstract*—The Internet has changed the way people access the information they need, and indeed how they live. Whether it is individuals reading emails or watching videos, or factories utilising automated fabrication devices, the access and processing of data is totally different. Thanks to the accessibility and the benefits that it brings into the lives, new research areas are emerging. One of the areas is Internet of Things (IoT) which connects countless of devices to the Internet. Increasing usage in IoT tremendously increases the count of connected devices to the Internet as well as the data generated and transferred. However, this increase brings several issues which could degrade the Quality of Service (QoS) with delays or even failed requests due to bandwidth limitations. Current tendency to solve problems that the Cloud Computing has is to perform computations close to the device as much as possible. This paradigm is called Edge Computing. There are several proposed architectures for the Edge Computing, but there is not an accepted standard by the community or the industry. Besides, there is not a common agreement on how Edge Computing architecture physically looks like. In this paper, we describe the Edge Computing, explain how its architecture seems, its requirements, and enablers. We also define an extensible, server architecture. The proposed Edge Server architecture has an ability to decide whether the task should be offloaded to the Cloud or to another Edge Server by considering the several parameters such as available resources and network delays. The resources of Edge Server can be extended with additional optional hardware or software modules to add new functionalities for artificial intelligence tasks, additional storage, wireless communication, etc. The server in the proposed architecture is also capable of performing real-time tasks and uses standard technologies to keep migration efforts at minimum. The paper also shows the results of an initial experiment, done without and with an Edge Server to compare computing performance.**

Keywords–Edge computing; real-time computing; edge computing requirements; enablers; Fog computing.

I. INTRODUCTION

Internet of Things (IoT) gave new possibilities and changed how people live their lives. Number of connected devices to the Internet is going up with the increased tendency towards IoT [1]. In 1992, "connected device" count was around one million which went up to 500 million in 2003 thanks to increased usage of personal computers. Later, IoT became even more popular and saw three billions of connected devices. In 2012, inclusion of of wearable devices increased this number to 8.7 billion. In 2013, this number went up to 11.2 billion owing to connected home appliances and in 2014, 14.4 billion with smart grids. The numbers increased in the upcoming years due to involvement of even small personal objects, such as toothbrushes, traffic lights, and table watches. Finally, even door levers are expected to be part of smart objects in 2020 [2].

Researches foresee that the connected devices are expected to be around 50 billion by 2020 [2][3]. This number is high as the Cyber-Physical Systems (CPS) and more intelligent components being used even for simple tasks. Tendency towards Cloud Computing and IoT devices leveraged the research in this domain and created new ones.

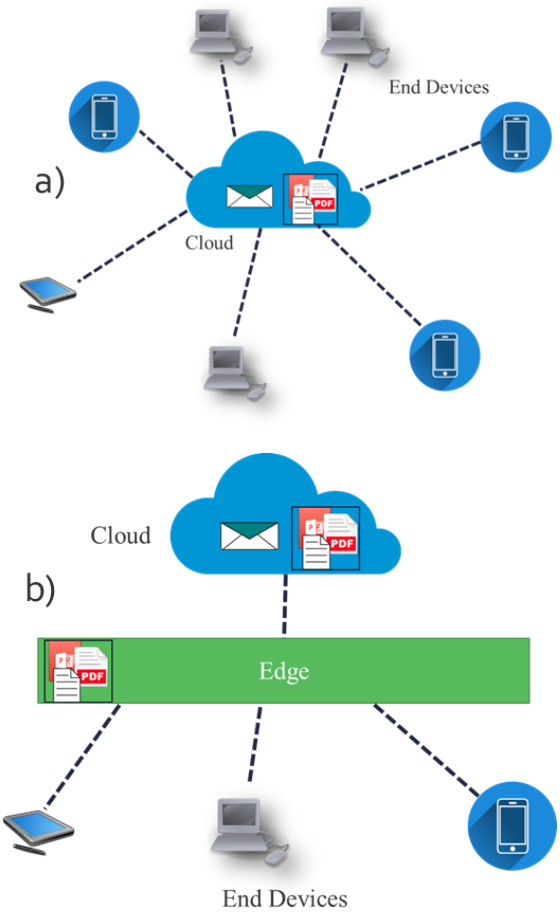

Figure 1. A simplified example showing the major difference between Cloud and Edge Computing. Cloud Computing (a) connects end-devices to the Cloud directly whereas Edge Computing (b) has an additional computing power in-between.

Cloud Computing or *the Cloud*, allows its users to store data, perform tasks using data centres through the Internet. The available resources in the Cloud granted low-powered or resource limited end-devices perform complex tasks in the Cloud, saving exceptional computational time [4]. Thanks to

the ubiquitousness of the Cloud, data can be accessed from anywhere and any time as long as an active Internet connection is available. Some everyday tasks such as checking e-mails, video streaming, photo browsing, and file sharing or industrial tasks such as getting sensor values or controlling robots are performed through the Cloud. Using different standards, a single infrastructure to keep the system reliable is becoming even more complex, causing difficult and costly maintenance. The companies and research institutes are working to avoid failures of tasks due to insufficient hardware and network resources. The physical distance to the Cloud and the available resources within the infrastructure increase the latency and reduce the Quality of Service (QoS). One of the recent paradigms in this area to solve issues of Cloud Computing is Edge Computing. Although there are several naming for Edge Computing such as Fog Computing and Cloudlets, within this paper, only the term *Edge Computing* will be used. Figure 1 shows the difference between Cloud and Edge Computing.

Cloud Computing [5] is an emerging technology which allows machines/people to access the data ubiquitously. It enables on-demand sharing of available computing and storage resource among its users which could be either human or machine, or even both. Today, it is even possible for a simple device to share its status or get information over Internet with millions of users.

A layer is a logical organisation of set of services, devices, or software with the same/similar specific functionality, mainly defined for abstraction of tasks. A tier is, however, a physical deployment of layers for scalability, security and to balance performance [6]. In Cloud Computing, communication between a device and the infrastructure which provides the service is direct, without involvement of other tiers. In Edge Computing, however, an intermediate component, or an Edge Server performs the initial computation.

Edge Computing combines multiple technologies such as Cloud Computing, Grid Computing, and IoT. It adds an additional tier between the Cloud and the end-devices and moves computational power to the end-device as close as possible. This means that, in the need of more computational resource by the end-device or a system, the task can be offloaded to an Edge Server instead of the Cloud. Edge Computing is expected to reduce the latency and increase the QoS for tasks which cannot be handled by these devices. These tasks are usually computationally heavy such as big data processing, video processing, artificial intelligence or time-sensitive. If the computation must be done in real-time, utilization of Cloud is out of the question since Cloud and Internet offers only best-effort service and delivery. A system is a real-time system only if it reacts to its environment by performing the correct predefined actions within the specified time intervals.

Real-time computing can be divided into three categories:

- *Hard Real-Time*: Failure in the system is mostly fatal. For example, if an airbag in a car deflates before or after the specified timeframe (between 100 ms and 300 ms, within 10 ms), it loses its protective impact [7].
- *Firm Real-Time*: A real-time category between hard and soft real-time. It tolerates some deadline misses, but increase in the misses degrades the service, in the end causing unacceptable results [8]. For example, miss-sorting colors of the parts are acceptable up to some point [9].
- *Soft Real-Time*: This category groups the real-time applications which are less critical and have wider deadline interval for their acceptance. For example, voice calls or video streams are tolerated in case some data packages are lost.

Some systems produce gigabytes of data per second [10][11]. Devices with limited computing capacity may also have critical deadlines for their primary task. In these situations, the task can be offloaded to an Edge Server using the same constraints and can be accomplished at this level. Depending on the outcome of the task, the system reacts to the result, e.g., sends the data back to the end-device.

Both Edge Computing and Cloud Computing are strongly related to IoT and allow accessibility of the data ubiquitously. To build an architecture, the issues on the current Cloud or IoT systems must be identified, requirements must be specified, enabling technologies must be listed, and then a concept must be given. Later, the concept can be implemented in an architecture, validated, and evaluated. This paper presents an ongoing work on Edge Computing with its clear description. It also explains its requirements and enablers to solve the introduced issues. The paper also shows an ongoing work to implement a novel server architecture which is capable of performing real-time tasks and take the necessary actions to provide a high QoS, such as offloading the task to another server or to the Cloud. The architecture will not simply be another architecture, but will compare the existing architectures and consider real-world requirements from the industrial use cases. The architecture will also be vendor-independent and extensible and it meet industrial requirements.

The rest of the paper is structured as follows: Section II introduces some of the existing work done and simulators in the area of Edge Computing. Section III describes the concepts and some definitions together with requirements and enablers. Section IV defines the Edge Server architecture to be implemented for Edge Computing in two sub-sections. Section V shows the initial experiment results with various scenarios and finally, Section VI concludes the paper and presents the future work.

II. Background

Although usage of the term "Edge Computing" is recent, there are already several proposed architectures available, each considering different aspects to meet the requirements of the Edge Computing.

The architecture proposed by IBM considers the requirements for autonomy and self-sufficiency of production sites. The architecture is three-layered to balance the workload between the Edge, the Plant, and the Enterprise. The challenges of the architecture are listed as productivity gains for high throughput, failure prevention for reliable system and high product quality, and flexibility while hiding the complexity and allowing reconfiguration without a lot of effort [12].

Another reference architecture is proposed by OpenFog Consortium [13]. This architecture names the core principles as pillars. Pillars group requirements within their scope. These pillars are Security, Scalability, Openness, Autonomy, Agility,

and Programmability. OpenFog Reference Architecture is proposed by covering industrial use cases.

Another recent initiative to build a common platform for Industrial IoT Edge Computing is EdgeX Foundry [14]. It was launched by Linux Foundation and initial contribution made by Dell. However, similar to OpenFog Consortium, it is also open for new memberships. EdgeX Foundry is a vendor-neutral open source software platform that interacts at the Edge of the network. It defines its requirements in architectural tenets as follows: platform agnostic in terms of hardware and operating system, flexible in terms of replaceability, augmentability, or scalability up and down, capable in storing or forwarding data, intelligent to deal with latency, bandwidth, and storage issues, secure, and easily manageable. A similar framework called Liota is being developed by VMware and it also aims at easy to use, install, and modify. Secondarily, it targets for a general, modular and enterprise-level quality. This framework is also open source and governed by VMware [15].

There are also several work done for computation and control in the Cloud. Below some of the related work is explained.

A research project called ”pICASSO” focuses on the control of a robot using a Cloud-based control platform. The project implemented a platform and a Cloud controller which can perform motion planning and control for industrial robots [16].

A recent work done by Givehchi, Imtiaz, Trsek, and Jasperneite [17] studies industrial use cases for using virtual control service in a private Cloud. Instead of using hardware programmable logic controllers (PLC) on site, they use a computer with multi-core processor and set each core as a virtual PLC to control sensors and actuators. The solution suggests a low-cost, but a slightly lower performance software PLC, compared to the hardware PLCs.

Another study on Cloud-based control is done by Goldschmidt et. al [9]. The work introduces a new architecture for scalable and multi-tenant Cloud-based control, virtualized PLCs. It also considers and evaluates the architecture with respect to its scheduling policies and time-sensitiveness. The Cloud architecture is located in a different physical location than the industrial site where the actual control is done and the communication is performed through Internet. The results showed over 99% success rate for tasks requiring response within one second. They suggest that the architecture is feasible for soft or firm real-time applications.

Realizing an unproven concept in real environments without testing and validating requires good investment of engineering time and money. However, using virtual environments which can simulate several hours of real environment tasks in couple of minutes save a lot of time.

CloudSim is a framework to model and simulate Cloud Computing infrastructures and their services. It supports modelling and simulation of large scale Cloud data centers, their application containers, costs as well as power consumption [18]. One simulation tool to evaluate the reliability of the system is called iFogSim and implemented by Gupta, Dastjerdi, Ghosh, and Buyya [19]. It is based on CloudSim and allows addition of fog or edge devices, creation of topologies and evaluation of resource management policies focusing on latencies [19]. Sonmez, Ozgovde, and Ersoy introduced another simulator called EdgeCloudSim [20]. It adds a mobility model and non-fixed delays into the network which is fixed in iFogSim. The simulator also gives detailed information on resource usage as well as the percentage of tasks statuses.

In both simulators, the data is passed to the Cloud in case there are no resources available in the Edge/Fog Server. However, in our scenario, the Edge Servers can also offload the tasks to other Edge Servers by considering the available resources, network and computation delays. Additionally, the end-devices do not have mobility, only the data does. We believe that there are no available simulators in the literature which can offload the tasks of immobile end-devices between the Edge Servers nor a standard Edge Server architecture which is capable of performing real-time calculations. The aim in this research is not simply to build another architecture, but to analyse the existing architectures and consider industrial requirements to make up a generic reference architecture which is vendor-independent and extensible. This ongoing work will build a novel architecture comparing the existing architectures, initially simulating in a virtual platform. To the best of our knowledge, this is not considered in any of the aforementioned reference architectures.

III. CONCEPT

Although Cloud Computing reduces costs of computation by saving hardware and giving flexibility, the physical distance to the device reduces the QoS. Additionally, if the resources of a Cloud infrastructure is shared, scheduling the tasks is a difficult task. Moreover, transmitting too much data to the Cloud more than a network can handle is unnecessary and causes network congestion [21]. If the task execution is critical and time-constrained, then an in-time correct reaction is necessary.

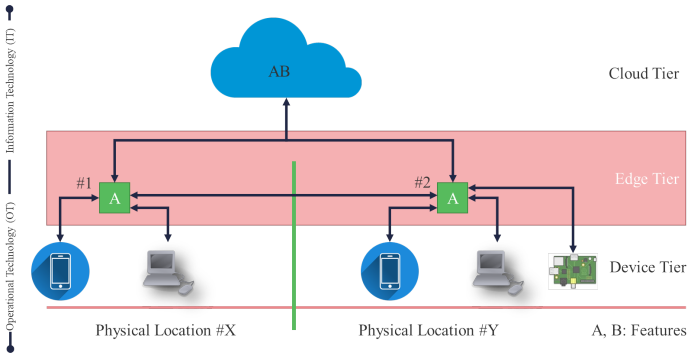

Figure 2. Simplified topology of the Edge Computing network.

One of the main goals of Edge Computing is to reduce latency and to keep the QoS as high as possible. As seen in Figure 1, in Cloud Computing, the Cloud infrastructure communicates with the end-devices directly. Edge Computing intends to solve the issues of Cloud Computing or IoT by adding an additional tier between the IoT devices and back-end infrastructure for computing and communication purposes. As depicted in Figure 2, this tier also has intermediate components for the first gathering, analysis, computation of the data. These intermediate components are called *Edge Servers*. Several architecture types for IoT-enabled applications are proposed

[22]. In this paper, a three-tier architecture is used. Unlike the example scenario of work done in [20], this paper assumes that the end-devices are not mobile.

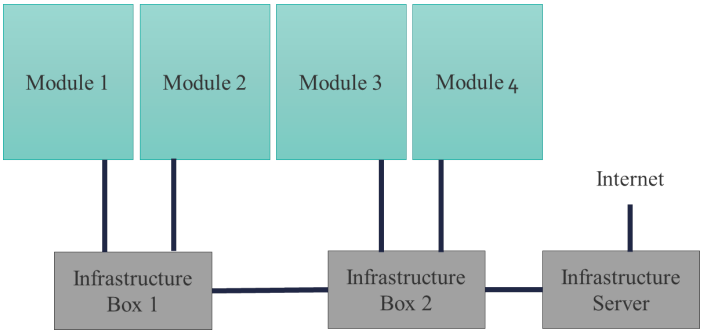

Figure 3. Overview of the modular testbed architecture to be used for validation and evaluation.

The scenario in our research involves a multi-vendor modular testbed for research purposes by *SmartFactory*^{KL} [23]. The testbed is composed of plug-and-produce modules and each of them performs one step of the production, independent from other modules. The modules are developed by different industrial partners and work together to produce a customizable and individualized product. As Figure 3 illustrates, the modules in the testbed are not directly communicating with each other, but through the infrastructure boxes. Each infrastructure box is connected with each other serially and provides pressured air, network connection, safety bus, and power to the modules. The communication to the Internet is performed through the central infrastructure server. The aim of the research is to add computing power into the infrastructure boxes to analyse, monitor the modules and react to the expected or unexpected situations, including real-time behaviour.

In our approach, we propose an extensible Edge Server model for Edge Computing to be integrated inside Infrastructure Boxes. If there are multiple Edge Servers in the same network, they are able to communicate with each other. Each server is orchestrated by itself, which means that they are not dependent of each other and aware of the neighbouring server capabilities in the same network. If a task cannot be guaranteed or performed by a server, the receiving server knows which other servers are capable of performing the same task. In all circumstances, the data transfer among devices will be performed through secured protocols. Figure 2 shows the simplified topology of a three-tier Edge Computing.

Edge Server is not a complete replacement of the Cloud with respect to its functionalities. Although its available resources are higher than the end-devices, they are lower than the Cloud. Instead, highly repeated tasks, or tasks that require in-time response are preferred to be executed in an Edge Server.

As seen in Figure 2, the proposed architecture for Edge Computing consists of *Cloud Tier*, *Edge Tier*, and *Device Tier*. In the Device Tier, there are end-user devices. The green blocks in the Edge Tier are Edge Servers. These servers gather, aggregate, analyse, and process the data before offloading them to the Cloud Tier or send back to the devices. The end-devices can be in the same physical location, or in different locations as depicted in the figure. When an end-device needs to communicate with the Cloud, first, the request is sent to the Edge Server which is at the closest location. Then, if the Edge Server is capable of completing the task by itself, it automatically handles the data and responds to the end-device with the result. If not, the data is offloaded to another server in the same tier provided that it exists. Otherwise, the data is offloaded to the Cloud. The decision process is made by considering available resources in other available servers in the same network, physical distance, and time requirements. In automation domain, Edge Tier can be seen as an edge or borderline between the Information Technology (IT) and Operational Technology (OT). In IT, the speed considerations are not critical whereas in OT, the communication or computing, or both must be real-time. Edge Tier isolates the network between IT and OT. Assume that *A* and *B* are features that could be serviced by the Cloud. For example, if a device in location *X* or *Y* needs the feature *B* to perform a task, the request will be orchestrated by the Edge Servers #1 or #2 and be sent to and performed by the Cloud. However, if, for example, the feature *A* is requested by an end-device in location *X*, first the Edge Server #1 will evaluate its own available resources. Depending on the urgency of the request, resource utilization, and calculated delays, it will either complete the request by itself or offload to the server #2 or to the Cloud.

Different tasks may have different priorities even though they are real-time. If there are multiple task requests, the server should pause the lower priority tasks while keeping track of the paused tasks or offloading them. The challenge here is to decide on the functionalities in the Edge Tier by keeping the costs at minimum and the QoS at maximum. However, deciding on the count and available resources of Edge Servers are also big challenges and big trade-offs. There are several aspects to consider before passing the data to the Cloud. To decide where to execute the task, each server has an orchestrator of which details will be explained in Section IV. According to this, the function should consider the priority of the task, resource utilization of the servers, computing cost for the task, and the physical distance or distance cost of the servers that is going to be used.

A. Requirements

Edge Computing is a paradigm which uses Cloud Computing technologies and gives more responsibilities to the Edge tier. These responsibilities are namely, computing offload, data caching/storage, data processing, service distribution, IoT management, security, and privacy protection [24].

Without limiting the Cloud Computing features, Edge Computing needs to have the following requirements, some of which are also defined for Cloud Computing [25][26]:

1) Interoperability: Servers in Edge Computing can connect with various devices and other servers. In Cloud Computing, IoT allows countless number of devices to communicate with humans or each other. This creates a big market for manufacturers of these devices. For this reason, there is the issue of interoperability with connected device using different communication protocols. Advanced Message Queuing Protocol (AMQP), Message Queue Telemetry Transport (MQTT), and TCP/IP are widely used and should be supported by Edge Computing. Using a widely-used and widely-known standard will remove the technology and language barriers, increasing interoperability among the devices.

2) Scalability: Similar to Cloud services, Edge Computing will also need to be adapted for the size of its users and sensors. Additional deployment of Edge Servers is costly and small number of Edge Servers is desirable in terms of economical aspects. For this reason, high scalability is also mandatory.

3) Extensibility: Computing technology is developing rapidly. After 2-3 years of deployment, clock speeds, memory size and program size increase, too. Easy deployment of new services and new devices with small effort is required for essential goal of Edge Computing. New functions and devices should be integrated without (re)configuration of the Edge network. Therefore, the system should allow extensibility with hardware and software components.

4) Abstraction: For the seamless control and communication, the abstraction of each Edge Node and group of nodes is required. Moreover, abstraction helps the topology of an Edge network to be flexible and reconfigurable. Fundamentally, an Edge node is located between device tier and Cloud tier. In other words, an Edge tier is a border between Information Technology (IT) and Operational Technology (OT). This tier can consist of one or more Edge nodes and groups. In this case, one Edge node of the group can share tasks or nodes in the group can be prioritized. Utilization of Application Programming Interfaces (APIs) in abstraction is useful to provide backward compatibility for the new functionalities or big changes in the architecture.

5) Time sensitiveness: Below OT, the operations may be near-real-time or real-time. Edge Computing is expected to solve time issues which Cloud computing cannot guarantee. Unlike Cloud Computing, physically close distance is one strength of reliable and fast communication without worrying about traffic problem. Video streaming service is one of expected applications of Edge Computing. It is required for real-timeness of the service provision. In addition, time-sensitiveness adds big benefits to the providers of reactive services, such as location-based advertisements and user-status based guide systems.

6) Security & Privacy: Using Cloud Computing services has a trade-off for enterprises like manufacturing and high-tech companies because there is a concern about the leakage of high knowledge and business activities outside their own organization. Edge Computing is a way to secure data contents, which is different from firewall which only controls external access into the network. It is also important to isolate the data by preventing access from even non-authorized users.

7) Reliability: Edge Servers provide real-time or non-real-time control for the devices. Real-time tasks may be vital which involve human safety. Therefore, it is vital to have a reliable system which reacts when it is needed and how it is needed. The physical reliability requirements for Edge servers providing services is similar to Cloud Computing. Harsh environments, such as factories and construction yards, require water-proof ceiling, fanless computers and dust-proof system. In power plant, magnetic shield is equipped by sensor gateways.

8) Intelligence: Multi-sensor generates tremendous amount of data and uploads into Cloud, directly. It causes network congestion and heavy load on the Cloud server. Edge Computing supports first and second filtering of these data by converting into higher level of data contents. Data filtering is implemented by rule-based engines or machine learning algorithms. In the case of multi-camera system like security systems, Edge Computing supports image processing, computer vision and enables object detection before transferring the data into the Cloud. Another example is predicting the failure or abnormalities in a production line by analysing the sensor data and taking the precautions for prevention or informing the user. These kinds of intelligent functions are necessary for Edge Computing.

9) Power: Unexpected shutdown or blackout is the cause of breakdown of Edge Server. Uninterruptible power supply (UPS) is required to give an ample amount of time to protect the electronic units and data storage in case of an unexpected shutdown due to power outage.

B. Enablers

Edge Computing uses wide range of technologies and brings them together. Within this domain, Edge Computing utilizes many technologies, such as wireless sensor networks (WSN), mobile data acquisition, mobile signature analysis, Fog/Grid Computing, distributed data operations, remote Cloud services, etc. Additionally, it combines the following protocols and terms:

1) 5G communication: It is the fifth generation wireless system which aims at higher capacity, lower power consumption, and lower latency compared to the previous generations. Due to increased amount of data between the data, 5G is expected to solve traffic issues which arose with the increased number of connected devices.

2) PLC protocols: Object Linking and Embedding for Process Control Unified Architecture (OPC-UA) is a protocol developed for industrial automation. Due to its openness and robustness, it is widely used by industries in the area of oil and gas, pharmaceutical, robotics, and manufacturing.

3) Message queue broker: MQTT and TCP/IP are popular message protocols of smart sensors and IoT devices. Supporting these message brokers, Edge Computing increases the device count that it connects. For the problem of MQTT security, AMQP is useful in the communication with Cloud Computing server.

4) Event processor: After messages of IoT arrive in the Edge server, event processor analyses those messages and creates semantic events using pre-defined rules. EsperNet, Apache Spark, and Flink are some examples for this enabler.

5) Virtualisation: Cloud services are deployed as virtual machines on a Cloud server or clusters. Using virtual machines allow running multiple instances of operating systems (OS) on the same server.

6) Hypervisor: As well as virtual machine, performance evaluation and data handling are required and realized by hypervisor to control virtual machines in the host computer.

7) OpenStack: Managing multiple resources could be challenging. OpenStack is a Cloud operating system that helps control of pools of computing and storage resources at ease through a control panel and monitoring tools.

8) AI platform: Rule-based engine and Machine learning platform supports data analysis in local level. As stated in Section III-A, this is quite important to reach one of the goals of Edge Computing which is to gather, analyse, and perform the first filtering of the data.

9) Docker: Virtual machines work with installation of operating systems. Unlike virtual machines, Docker is a Container as a Service (CaaS), which can use a single shared operating system and run software in isolated environment. It only requires the libraries of the software which makes it a lightweight system without worrying about where the software is deployed.

IV. Architecture Design

An Edge Server must be capable of gathering, aggregating the data, computing and transferring it back to the end-device. However, in the meantime, the servers must be able to communicate with other Edge Servers within the network, in case their resources are not enough to perform the task. Alternatively, they must be able to offload the data to the Cloud. In other words, the Edge Servers must have a reliable and communicable network between each other and the Cloud.

For seamless task handling and communication, the servers must follow some standards compatible with each other. This is not a simple task, since this technology contains several aspects to consider such as resource allocation, scheduling, scaling, storage, etc. The architecture must also be able to handle time-critical or real-time tasks. The software must also be designed or modified to work with the real-time capable system [27]. Last but not least, the server must be extensible with plug-and-play modules to advance or add new functionalities via hardware or software modules. These extensions must be validated before usage, to keep the system functional and to prevent intrusion. Such architecture design is divided into Hardware Modules and Software Components which are explained in the next subsections.

A. Hardware Modules

As mentioned in the previous sections, to solve the problems of Cloud Computing, Edge Servers or "Edge Nodes" need to have more computing power than the end-devices. However, the hardware in the Edge Tier is also limited compared to the Cloud. Therefore, to keep the balance between the performance and costs, first, the use cases for the Edge Servers must be defined, then, the resource must be considered to handle these defined use cases in-time. During the research, it has been decided to use Samsung Artik 710 as the hardware. It has high performance 8-core 64-bit ARM Processor, integrated wireless adapters, 1 GB RAM and 4 GB flash memory, extensible with an SD card. The price to performance ratio, internal real-time clock (RTC), and availability of the open source repository also played a big role for this decision [28][29].

The data produced by the end-devices are not directly sent to the Cloud or back-end infrastructure, but initial computing is performed on these servers. Considering the number of connected devices and the data they produced, these servers are used to aggregate, analyse, and process the data before sending it into the upper layer, the infrastructure, or back to the device.

The proposed Edge Server architecture is to be designed modular and should provide functionalities for real-time and non-real-time control, as well as real-time communication. Each server in our proposal has a *Core Node* of which functionalities can be extended by plugging additional optional modules in. Figure 4 shows an overview of possible modules or devices that could be attached to the core node. Hardware

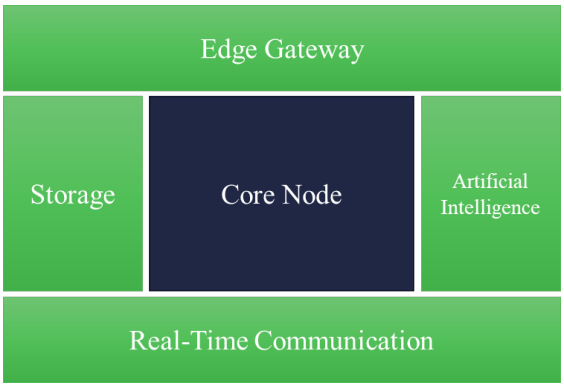

Figure 4. View of the proposed extensible Edge server architecture with its major functionalities, where green blocks extend the functionalities for the blue core node.

modules can also have their computing power, or simply improve the available functionality of the core node. For example, to capture big data and store, a storage module can be attached or in the case that machine learning algorithms are desired to be executed on the server, a dedicated artificial intelligence (AI) module with dedicated Graphics Processing Unit (GPU) can be connected. This module will be available for use with none to minimal configuration.

To preserve the integrity of the system and prevent unauthorized hardware from breaking the system functionality, only verified hardware must be allowed for connection. Hardware extensions are planned to be made using a physical master key, which is thought to be a USB device. Whenever this device is plugged in, the Edge Server will be ready to identify and allow new hardware to be added.

As mentioned in Section III-A, scalability is quite important to accomplish the tasks. In the scope of scalability, one server is expected to be aware of its neighbouring servers along with their functionalities. When a server is plugged into the network and turned on, first, it publishes that it is available. Then, it publishes its resources along with the available functionalities inside. Using the previous example, in case an AI module is connected to one server, other servers are informed with this functionality and they can utilize this server more often for AI-related tasks. The decision, of course, depends on the conditions required by the task, such as deadline.

Core Node should support multiple communication interfaces. Therefore, time and speed considerations are quite important for choosing the best hardware. The Edge Server is also expected to perform real-time computing and control. Therefore, reliability and stability of the hardware is also mandatory.

B. Software Components

As mentioned in Section IV-A, the server has a *Core Node* which is capable of performing real-time tasks. Before implementing the software architecture of such system, it is important to define the roles of software, clarify, and assign separate roles to the components. Rather than choosing the technologies or languages, which the software components are going to be implemented in, all components defined here can be implemented in any language as long as they satisfy the

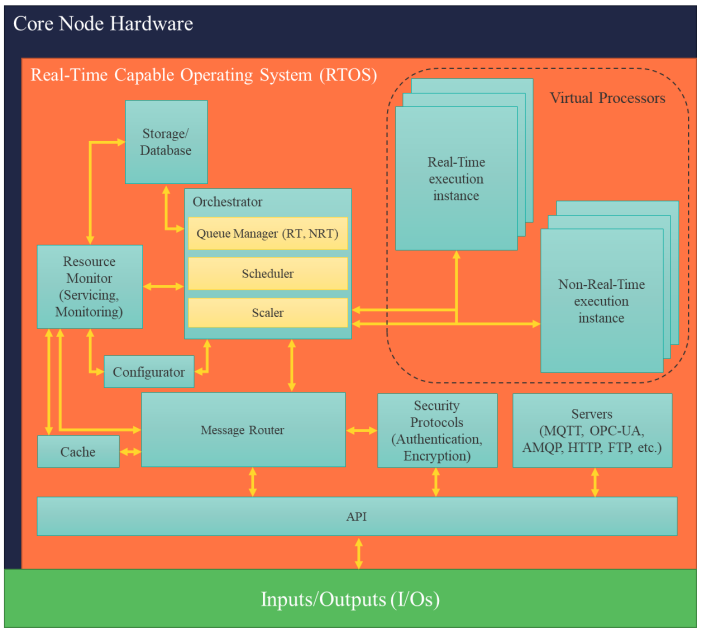

Figure 5. Proposed Software Architecture of Core Node.

requirements. Time-sensitiveness requires implementation of several components which are compatible with each other. These components should enable a reliable, stable in-time response and take correct actions.

Figure 5 shows the required software components inside the *Core Node* to fulfil the requirements. Below, these components are explained:

Real-Time Capable Operating System (RTOS): To implement software components, the operating system (OS) in the hardware must also be real-time capable. There are many OSes which can handle real-time tasks. However, the choice of the OS should be made by considering its available support, availability of the source code or openness for modifications, and applicability of the OS into the chosen hardware platform. Having a real-time capable kernel does not mean that the system will work in real-time. Applications, APIs, and the system must be designed properly to benefit from real-time functionality [30]. It should have a native support for the chosen *Core Node* hardware that it is running on. With the current stage of the research, the operating system has been decided to be based on Linux kernel, as it is open source and modification is easy with the plenty of resources available. The chosen Samsung Artik 710 hardware comes with pre-installed Fedora 24. However, the out-of-the-box version is not real-time capable. Therefore, the real-time patch [31] will be integrated into the kernel and it will be recompiled. Nevertheless, the other software components will be OS-neutral, therefore could be adapted to other OSes.

Inputs/Output (I/Os)s: I/Os are the interfaces which connect the hardware with the software. These are also used to connect other physical modules with the core node such as Edge Gateway for real-time communication.

API: APIs will be used for all communication with the I/Os, the end-devices, or the Cloud. An API is necessary to abstract the functionalities of other components. It allows internal modifications in case a new software component added without requiring complete change in the system. It also guarantees that the requests cannot interfere with the internal components since direct access to the individual modules or components is not allowed. Another goal of the architecture is to keep the migration efforts at minimum. Therefore, it is important to choose an accepted and standard language for the API to make it compatible with as many device and software as possible. Another aspect to consider is to choose a lightweight, yet stable API as a low latency is desired. Last but not least, the API should make sure that the requests are always authorized. This is performed by evaluating the request with *Security Protocols* component.

Message Router: As soon as the task or data arrives, this component retrieves and routes it to the location where task should be handled by communicating with *Resource Monitor* component. In case there are no resources available in this server, the task will either be transferred to another Edge Server or to the Cloud. This component always makes sure that the incoming task is from a trusted device by interacting with the *Security Protocols* component.

Configurator: The server and their modules are automatically configured as soon as they are attached. Nevertheless, their manual configuration or tweaks are performed via this component. It provides a Web-based and shell-based administrator panel to modify server properties, monitor the status, perform low-level resource allocations, and adjust orchestrator parameters, etc. Additionally, this component detects other nearby servers in the same network and configures the server to use them, when necessary. Similar to other components, this component is also accessed through the API.

Storage/Database: This component is used to store temporary data for the active or waiting tasks. However, the component will not keep the permanent data of the tasks. To keep the permanent data, storage module or the Cloud is recommended.

Servers: Standalone servers which are used out-of-the-box with only configuration changes are encapsulated in this component. The servers enable API communication as well as internal communication among the components. The servers are configured through the API via *Configurator* component.

Security Protocols: One of the most important requirement for the Edge Computing is keeping the data secure and private. Security itself is a vast aspect to consider. Therefore, a dedicated component to handle all security-related issues is necessary. This component monitors all incoming connections and takes the necessary actions in case the request is unexpected or unauthorized. Since the Edge Server will be accessible via the Cloud, the component is also responsible to prevent Internet-based attacks. Moreover, the component makes sure that the data privacy is preserved by encrypting, decrypting the data and issuing and exchanging keys, etc.

Resource Monitor (RM): The computing power in the Edge Server hardware is limited. RM actively monitors all available resources of Edge Servers in the network. It is also aware of all the other connected Edge Servers and their attached modules. RM directly interacts with the *Message Router* and decides whether the task received must be processed in this server or offloaded to another location. If the task is to be executed in another server, this component informs the

Message Router and points the target without further action. This decision is made by bi-directional communication with *Orchestrator*.

Cache: A temporary storage component to serve the data faster for the future requests. It is especially used when *Resource Monitor* decides that the task is not going to be executed in the current server.

Virtual Processors: For performance and power reasons, it is typical to have multi-core hardware. In multi-core hardware, one thread is generally executed on a single core. If a software is not optimized for multi-core, it cannot benefit from multi-core hardware. On the other hand, multi-threaded software execution is distributed among the cores. However, in either case, it is possible to set central processing unit (CPU) affinity of a process, that is a running instance of the software or program. Low-level programming makes configuration of the kernel possible to add virtual processors, limit or specify the available resources, and assign specific processes to these virtual processors.

Orchestrator: If RM allows execution of the task in this server, this component becomes active. Using event-based communication with the RM, the task is handled according to its urgency or the priority. If there is enough resource to execute the task, the task is immediately executed. If not, determined by the availability of the resource, task can be handled in different ways using the sub-components. This component has three different sub-components, namely: *Scheduler*, *Scaler*, and *Queue Manager*.

Scheduler: If a task is chosen to be executed in this server, it must be carefully scheduled to avoid deadline misses, especially for real-time tasks. Although multi-threaded software is usually scheduled by the OS and assigned to the CPU cores, the affinities may need to be adjusted. Depending on urgency or the priority of the new task, this component is responsible to set CPU affinities for the running tasks taking their priorities into consideration. The scheduler should be designed to minimize the waiting time of paused tasks.

Scaler: Scheduler sorts the execution times of the tasks. In the proposed architecture, some of the cores are dedicated for real-time tasks. Tasks not optimized for multi-core systems are by default assigned to run on a single core. If *Scheduler* is not able to meet the deadlines of the critical tasks, this component can increase the available core count for the real-time tasks to have them run on multiple cores, even if the cores are not assigned to execute real-time tasks. Of course, this is only possible provided that the tasks are multi-threaded.

Queue Manager: If a task cannot be executed immediately, one other possibility is to queue it. The queue contains both real-time (RT) and non-real-time (NRT) tasks. This component communicates with the *Scheduler*, and stores the tasks marked to pause and forwards the paused tasks to scheduler, following a variable scheduling algorithm.

V. Experiment Results on Different Hardware

SmartFactory^{KL} concept was depicted in Figure 3 in Section III. It was also mentioned that, the factory comprises multiple modules from different vendors. Figure 6a shows the the modules, infrastructure boxes and the infrastructure server as they are deployed in real world. Figure 6b, shows one of the modules that is responsible for one task during production.

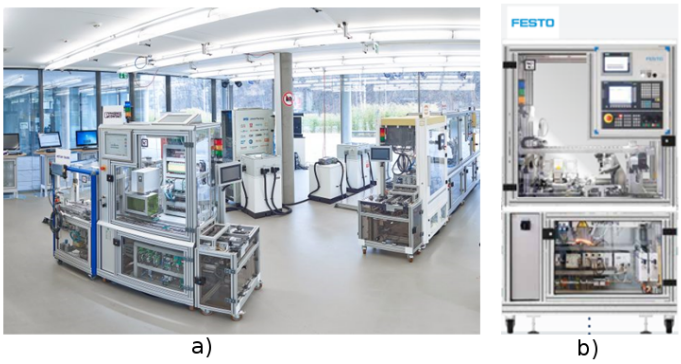

Figure 6. a) *SmartFactory*^{KL} with all modules, infrastructure boxes and the infrastructure server, b) Close-up view of one of the modules.

Edge Server will be used in several real world scenarios such as object detection, production priority change, emergency stop and production cancellation. As a first experiment, we chose object detection use case in which we can analyze the objects using a smart glass. The experiment does not have a real-time dependency, therefore they were made with best effort approach. The experiment exposes an on-site Edge Server to improve the overall performance and realize new features which are not applicable in Cloud-computing.

It compares Edge Computing-based service with the only low-end device-based computing. Comparison with various combination of hardware is the experiment test coming from the problem of Cloud-based computing services. The test shows the advantage of using Edge Computing and help find the conditions where Edge Computing is better than Cloud or low-end device-based computing. The purpose of this experiment is to prove the improvement that Edge Computing configuration shows better performance than legacy configuration or Cloud-based system configuration. The experiment contains two computing servers which are CPS modules located in front of a machine itself and Edge Server in order to share the computing load. The test was made using five different network configurations. The configurations are divided into 1) CPS module alone, 2) Edge Server alone, 3) Connection with Edge server via wired communication, 4) Connection via wired and wireless communication, and 5) Connection via two different wireless communication. The input is video streaming data transferring the objects and people in front of the camera, attached in the production modules.

As seen in the Table I, we found the following results through the tests. Using CPS module with an Edge Server performed four times faster object detection than the CPS module alone. All kinds of connection types satisfied the bandwidth requirements of single connection with CPS module. Even if the wireless connection was 400 Mbps, the image streaming data could be transferred without any delay. Faster wireless connection will be more useful when there are more data being transferred in order to keep low latency. CPS test did not show any delay but the computation time was long since no Cloud services are used and computing power is weak. Edge Server alone showed the best results, however needed more memory because of the need to run additional software to test the performance.

Test Configuration	CPS Module	Comm. Method	Edge Computing
CPS module alone without Edge service (Raspberry Pi 3: Quad core 1.2GHz 64bit CPU and 1 GB RAM)	U: 4.6 to 4.83s C: 22 - 67% M: <360 KB (36%)	Not used	Not used
Edge Server alone (Intel I7 64bit 16GB RAM NVIDIA GTX 950)	Not used	Not used	U: 1.2 to 1.40s C: 28 - 51% M: <550∼650 MB G: <6∼16% (210-290MB)
CPS module communicating with Edge Server via Wired connection	Only transferring camera images	1 Gbps Wired Connection	U: 1-2s C: 16-23% M: 250-310 MB G: 6-17% (571MB)
CPS module communicating via 802.11ngb with Edge Server with the router via Wired connection	Only transferring camera images	802.11ngb @0.4 Gbps and 1 Gbps wired connection	U: 1-2s C: 17-23% M: 250-310 MB G: 6-14% (570MB)
CPS module communicating via 802.11ngb with Edge Server with the router via Wireless connection	Only transferring camera images	802.11ngb @0.4Gbps and 802.11ac @1.2Gbps	U: 1-2s C: 17-23% M: 250-310 MB G: 6-14% (570MB)

TABLE I. Object detection experiment results with different configurations (U: Update period (Average (AVG)), C: CPU Util. AVG, M: Memory Usage, G: Graphical Memory Load).

VI. Conclusion and Future Work

Edge Computing is a recent term which moves the services from the Cloud to the device as close as possible and open for new innovations. It is a borderline between the Cloud and the device tier. Although the Cloud Computing and IoT have brought many advantages in the previous years, increased number in the connected devices raised some issues, such as latency and low QoS problems. Edge Computing is believed to solve these issues by analysing the issues and considering the requirements of real world use cases.

There are already several existing proposed architectures in the domain of Edge Computing, such as EdgeX Foundry, Liota, and OpenFog Reference Architecture. Although they are also extensible and they allow inter-connectivity, they do not focus on the real-timeliness of the architectures.

This paper showed an ongoing work on how *Edge Computing* physically looks like together with its requirements and enablers. It also explained the basics on how the communication between the end-devices and Edge Servers are expected to be. Last, the paper proposed an ongoing work for Edge Server architecture which is capable of performing real-time computations. The servers are able to orchestrate the tasks and find the best host to offload the requested task by an end-device. The offloading can be done between other Edge Servers in the network, or the Cloud. The work is being developed by considering the real-world use cases of the industrial partners. The first experiment is made using object detection algorithms with intermediate devices, without real-time requirement. However, further experiments will be performed with industrial use cases requiring real-time deadlines. Later on, the comparison with the legacy systems will be made.

One ongoing task is implementation of a simulator similar to CloudSim framework, including all components explained in Section IV. This will help us find out the optimal parameters for the hardware modules and the software components. Current version of the simulator is capable of emulating the computer central processing unit (CPU), generating tasks with specific properties, and scheduling them with feasibility checks. Later, the architecture will be implemented, validated and evaluated on an optimal hardware chosen for the work.

In the proposed solution, when a task execution started, it can be performed only using one server. Another future work is to divide these tasks into multiple machines to exploit the resource utilization of the available resources, which is called task migration. This work could further be extended by using artificial intelligence instead of mathematical calculation for optimizations. For example, similar tasks can be grouped, and the historical data can be used to estimate the task duration and the action can be taken.

Acknowledgment

This research was funded in part by the H2020 program of European Union, project number 723094 (project FAR-EDGE). The responsibility for this publication lies with the authors. The project details can be found under project website at: `http://www.far-edge.eu`

References

[1] V. Gezer, J. Um, and M. Ruskowski, "An extensible edge computing architecture: Definition, requirements and enablers," in Eleventh International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM-2017), Special Session on Edge Computing, International Academy, Research, and Industry Association (IARIA). IARIA, November 2017. [Online]. Available: https://www.researchgate.net/publication/321134141_An_Extensible_Edge_Computing_Architecture_Definition_Requirements_and_Enablers

[2] NCTA, "The Growth of The Internet of Things," Infographic, May 2014, [retrieved: May 2018]. [Online]. Available: https://www.ncta.com/platform/industry-news/infographic-the-growth-of-the-internet-of-things/

[3] D. Evans, "The Internet of Things - Cisco," Cisco, White Paper, April 2011, [retrieved: May 2018]. [Online]. Available: https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[4] H. H. Holm, J. M. Hjelmervik, and V. Gezer, "CloudFlow - an infrastructure for engineering workflows in the cloud," in UBICOMM 2016: The Tenth International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies. IARIA, October 2016, pp. 158–165.

[5] P. M. Mell and T. Grance, "The nist definition of cloud computing," in National Institute of Standards and Technology Technical report, September 2011, [retrieved: May 2018]. [Online]. Available: https://www.nist.gov/publications/nist-definition-cloud-computing?pub_id=909616

[6] R. Lhotka, "Should all apps be n-tier?" Blog, 2005, [retrieved: May 2018]. [Online]. Available: http://www.lhotka.net/weblog/ShouldAllAppsBeNtier.aspx

[7] E. Olderog and H. Dierks, Real-Time Systems: Formal Specification and Automatic Verification. Cambridge University Press, 2008.

[8] T. Kaldewey, C. Lin, and S. Brandt, "Firm real-time processing in an integrated real-time system," Universitry of York, Department of Computer Science - Report, vol. 398, 2006, p. 5.

[9] T. Goldschmidt, M. K. Murugaiah, and C. Sonntag, "Cloud-Based Control: A Multi-Tenant, Horizontally Scalable Soft-PLC," in IEEE 8th International Conference on Cloud Computing, 2015.

[10] S. Higginbotham, "Sensor Networks Top Social Networks for Big Data," Article, 2010, [retrieved: May 2018]. [Online]. Available: https://gigaom.com/2010/09/13/sensor-networks-top-social-networks-for-big-data-2/

[11] T. Valich, "Big Data In Planes: New P&W Gtf Engine Telemetry To Generate 10GB/s," Article, 2015, [retrieved: May 2018]. [Online]. Available: https://vrworld.com/2015/05/08/big-data-in-planes-new-pw-gtf-engine-telemetry-to-generate-10gbs/

[12] I. C. A. Center, "IBM: Internet of Things," Cloud Garage Method, 2017, [retrieved: Sep 2017]. [Online]. Available: https://www.ibm.com/devops/method/content/architecture/iotArchitecture/industrie_40

[13] "OpenFog Consortium Reference Architecture," Website, 2017, [retrieved: May 2018]. [Online]. Available: https://www.openfogconsortium.org/ra/

[14] "EdgeX Foundry Architectural Tenets," EdgeX Foundry Wiki, 2017, [retrieved: May 2018]. [Online]. Available: https://wiki.edgexfoundry.org/display/FA/Introduction+to+EdgeX+Foundry

[15] "VMware Introduces Liota," Website, 2017, [retrieved: May 2018]. [Online]. Available: https://www.vmware.com/radius/vmware-introduces-liota-iot-developers-dream/

[16] "pICASSO Project," Website (German), [retrieved: May 2018]. [Online]. Available: https://www.projekt-picasso.de/projekt/

[17] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite, "Control-as-a-service from the cloud: A case study for using virtualized plcs," in 2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014), May 2014, pp. 1–4.

[18] "The Internet of Things - Cisco," CloudSim Website, [retrieved: May 2018]. [Online]. Available: http://www.cloudbus.org/cloudsim/

[19] G. H., A. V. Dastjerdi, S. K. Ghost, and R. Buyya, "iFogSim: A Toolkit for Modeling and Simulation of Resource Management Techniques in Internet of Things, Edge and Fog Computing Environments," in Software Practive and Experience, June 2016.

[20] C. Sonmez, A. Ozgovde, and C. Ersoy, "Edgecloudsim: An environment for performance evaluation of edge computing systems," in 2017 Second International Conference on Fog and Mobile Edge Computing (FMEC), May 2017, pp. 39–44.

[21] H. Al-Bahadili, Simulation in Computer Network Design and Modeling: Use and Analysis: Use and Analysis, ser. Premier reference source. Information Science Reference, 2012.

[22] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," IEEE Communications Surveys Tutorials, vol. 17, no. 4, 2015, pp. 2347–2376.

[23] D. Zuehlke, "Smartfactory — towards a factory-of-things," vol. 34, no. 1, 2010, pp. 129 – 138. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578810000143

[24] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and Challenges," IEEE Internet of Things Journal, vol. 3, no. 5, October 2016, pp. 637–646.

[25] G. Orsini, D. Bade, and W. Lamersdorf, "Context-Aware Computation Offloading for Mobile Cloud Computing: Requirements Analysis, Survey and Design Guideline," Procedia Computer Science, vol. 56(1), December 2015, pp. 10–17.

[26] J. Shamsi, M. A. Khojaye, and M. A. Qasmi, "Data-intensive cloud computing: Requirements, expectations, challenges, and solutions," Journal of Grid Computing, vol. 11, no. 2, Jun 2013, pp. 281–310.

[27] S. Rostedt, "Intro to Real-Time Linux for Embedded Developers," Linux Foundation Blog, 2013, [retrieved: May 2018]. [Online]. Available: https://www.linuxfoundation.org/blog/intro-to-real-time-linux-for-embedded-developers/

[28] "Samsung ARTIK IoT Platform - ARTIK 710 IoT module," Samsung IoT Website, [retrieved: May 2018]. [Online]. Available: https://www.artik.io/modules/artik-710/

[29] "Samsung ARTIK Kernel on GitHub," Github, [retrieved: May 2018]. [Online]. Available: https://github.com/SamsungARTIK/linux-artik/

[30] J. Huang, "RTMux: A Thin Multiplexer to Provide Hard Realtime Applications for Linux," Embedded Linux Conference Europe, October 2014, [retrieved: May 2018]. [Online]. Available: https://events.linuxfoundation.org/sites/events/files/slides/rtmux_1.pdf

[31] L. K. Projects, "RTLinux Repository," Website, 2007, [retrieved: May 2018]. [Online]. Available: https://www.kernel.org/pub/linux/kernel/projects/rt/