International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

An Extensible Real-Time Capable Server Architecture for Edge Computing

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

***Abstract*—Internet changed how the people live and access the information that they need. 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 through the Internet. However, this increase brings several issues which could degrade the Quality of Service (QoS) with such as delays or even failed requests. Edge Computing is a new paradigm which is believed to solve the problems that the current IoT and Cloud solutions have. This paper introduces a new architecture for Edge Servers which is expected to reduce the latency and solve the resource problems of the end-devices. The proposed Edge Server architecture has an ability to decide whether the task should be offloaded to the Cloud or another Edge Server by considering the several parameters such as available resources and network delays. The server is also real-time capable and extensible with optional modules such as artificial intelligence, storage, wireless communication, etc.**

***Keywords*—Edge computing, fog computing, edge server, real-time computing**

I. Introduction

Internet of Things (IoT) gave new possibilities and changed how people live their lives. Tendency towards Cloud Computing and IoT devices leveraged the research in this domain and created new ones. 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 let low-powered or resource limited end-devices perform complex tasks in the Cloud, saving exceptional computational time [1]. Thanks to ubiquitousness of the Cloud, the data can be accessed from anywhere and anytime as long as an active Internet connection is available. With usage of the Cloud in daily tasks and used devices in the area, it is expected to have 50 billion devices to the network [2][3]. This growth in the count of connected devices also boosts the generated data, tremendously stressing the infrastructure. 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 or Cloudlets, within this paper, only the term *Edge Computing* will be used. Fig. 1 shows the difference between Cloud and Edge Computing. In Cloud Computing, the end-devices are connected to the Cloud directly, without an intermediate computing component. In Edge Computing, however, an intermediate component 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 Comp-

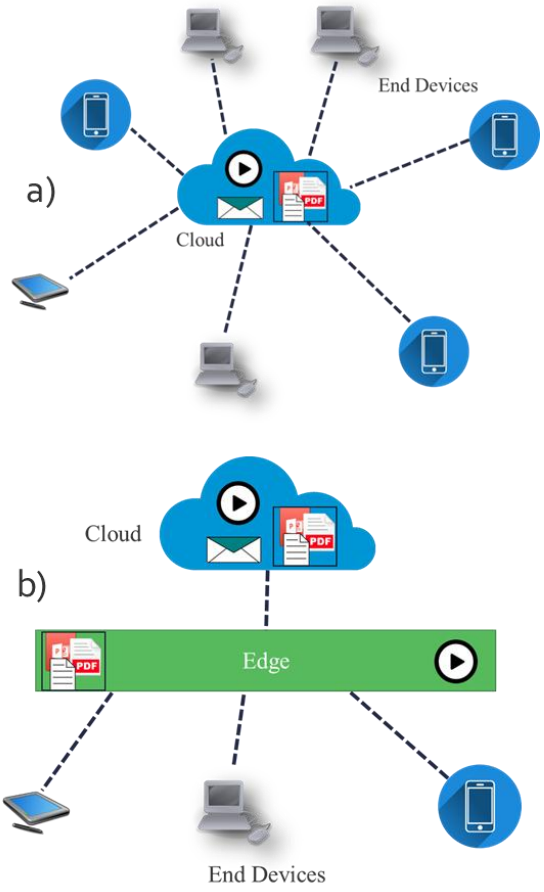

Figure 1. A simplified example showing the major difference between Cloud and Edge Computing. Cloud Computing (a) connects end-devices to the Cloud directly whereas Edge Computing (b) has an additional computing power in-between.

uting 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. In *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 [4]. *Firm Real-Time* is 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 [5]. For example, miss-sorting colors of the parts are acceptable up to some point [6]. *Soft Real-Time* 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. Devices with limited computing capacity may offload the tasks to an Edge Server using the same constraints and can be accomplished at this level.

This paper presents an ongoing work to implement an Edge Server architecture which is capable of performing real-time tasks and take the necessary actions to provide a high QoS. 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 aim is to build a vendor-independent an extensible architecture while meeting the 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 scenario and the approach. Section IV defines the architecture to be implemented as Edge Computing solution in Edge Server in two sub-sections and finally Section V concludes the paper and presents the future work.

II. Background

There are several work done for computation and control in the Cloud and 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 [9].

A recent work done by Givehchi, Imtiaz, Trsek, and Jasperneite [8] 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 [6]. 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 [9]. One simulation tool to evaluate the reliability of the system is called iFogSim and implemented by Gupta, Dastjerdi, Ghosh, and Buyya [10]. 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. Sonmez, Ozgovde, and Ersoy introduced another simulator called EdgeCloudSim [11]. 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 neither 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. This ongoing work will build a novel architecture comparing the existing architectures, initially simulating in a virtual platform.

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 are 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 [12]. If the task execution is critical and time-constrained, then an in-time correct reaction is necessary.

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

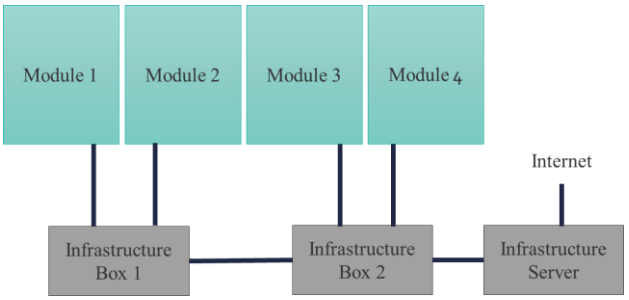

Figure 2. Overview of the modular testbed architecture to be used for validation and evaluation.

Unlike the work done in [11], in this paper, the end-devices are not mobile. The scenario in our research involves a multi-vendor modular testbed for research purposes by *SmartFactory*^{KL} [13]. These modules are plug-and-produce modules and each of them performs one step of the production independent of other modules. As Fig. 2 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 distributed and aware of the neighboring server capabilities in the same network. If a task cannot be guaranteed/performed by a server, the receiving server knows which other servers are capable of performing the same task. Fig. 3 shows a simplified topology of a three-tier Edge Computing.

As seen from the Fig. 3, an Edge Server is not a complete replacement of the Cloud with respect to its functionalities. Instead, highly repeated tasks, or tasks that require in-time response are preferred to be executed in an Edge Server. 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.

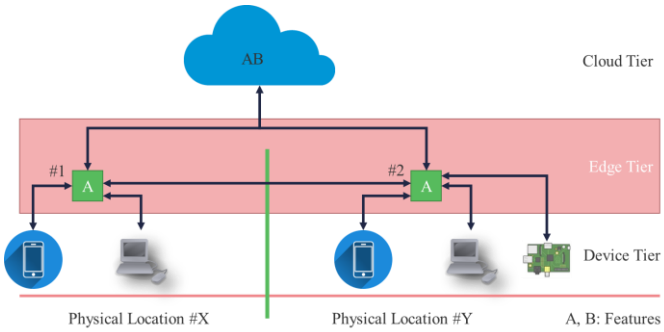

Figure 3. Simplified topology of the Edge Computing network.

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. To decide where to execute the task, each server has an orchestrator of which details will be explained in Section IV.

IV. Architecture

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 the other Edge Servers within the network in case their resources are not enough to perform the task or 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 a 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 so many 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 [14]. 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. Such architecture design is divided into Hardware Modules and Software Components which are explained in the next subsections.

A. *Hardware Modules*

Edge Computing should be able to reduce the latency and the load in the Cloud while keeping the QoS as high as possible. Edge Nodes, or Edge Servers 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 costs low, 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. Each server in our proposal has a *Core Node* of which functionalities can be extended by plugging additional optional modules in. Fig. 4 shows an overview of possible

International Journal of Advance in Computer Science & its application – IJCSIA 2018
Copyright © Institute of Research Engineers and Doctors , SEEK Digital Library
Volume 7 : Issue 2 [ISSN : 2250-3765] - Publication Date: 25 June , 2018

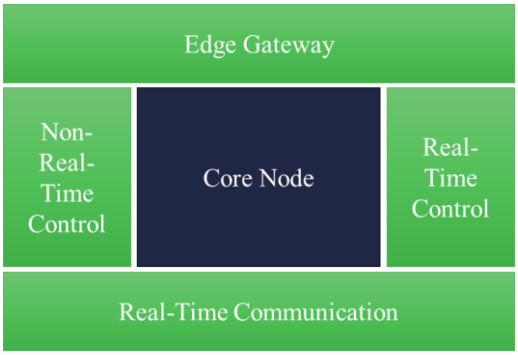

Figure 4. An extensible Edge Server concept showing its modules and devices which add functionalities.

modules or devices that could be attached to the core node. Hardware 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. On the other hand, the core node can still have enough storage perform small tasks.

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. All components defined here can be implemented in any language as long as they satisfy the 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. Fig. 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. Having a real-time capable kernel does not mean that the system will work in real-time. Applications, application programming interfaces (APIs), and the system must be designed properly to benefit from real-time functionality [15].

Inputs/Outputs (I/Os): 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

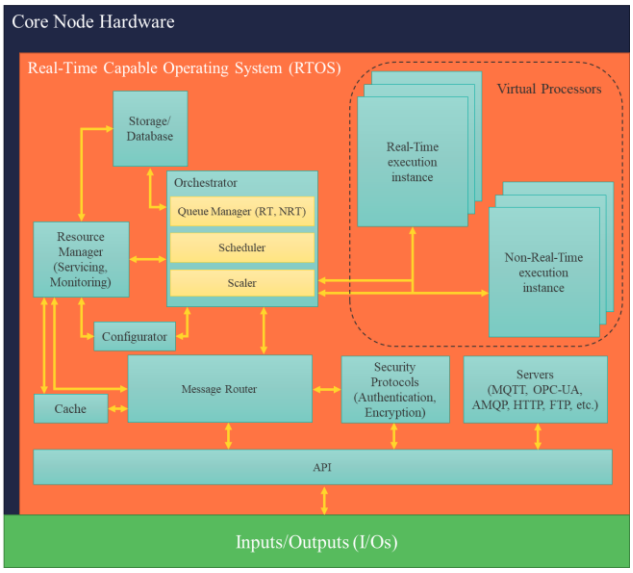

Figure 5. Proposed Software Architecture of Core Node.

components since direct access to the individual modules or components is not allowed.

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 Manager* 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.

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.

Storage/Database: This component is used to store temporary data for the active or waiting tasks.

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.

Security Protocols: One of the most important requirements 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.

Resource Manager (RM): The computing power in the Edge Server hardware is limited. RM or *Observer* 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.

Cache: A temporary storage component to serve the data

faster for the future requests. It is especially used when *Resource Manager* 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 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, which 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. 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. This component is responsible to set CPU affinities for the running tasks taking their priorities into consideration.

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.

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 is responsible for queuing the tasks and resuming them.

v. Conclusion and Future Work

Edge Computing is a recent technology and open for new innovations. It is believed that Edge Computing will solve the issues that the high usage of IoT and the Cloud solutions have. In this paper, we proposed an ongoing work for Edge Server architecture which is capable of performing real-time computations. The architecture is 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. One future task is implementation of a simulator based on 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. Later, the architecture will be implemented, validated and evaluated on an optimal hardware chosen for the work. Another future work is to divide these tasks into multiple machines to exploit the resource utilization of the available resources. This work could further be extended by using artificial intelligence instead of mathematical calculation for optimizations.

Acknowledgements

This research was funded in part by the H2020 program of European Union, project number (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] 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.

[2] NCTA, "THE GROWTH OF THE INTERNET OF THINGS," Infographic, May 2014, [retrieved: Sep 2017]. [Online]. Available: https://www.ncta.com/platform/

[3] D. Evans, "The Internet of Things - Cisco," Cisco, White Paper, April 2011, [retrieved: Sep 2017]. [Online]. Available: https://www.cisco.com/c/dam/en us/about/ac79/docs/innov/IoT IBSG 0411FINAL.pdf

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

[5] T. Kaldewey, C. Lin, and S. Brandt, "Firm real-time processing inan integrated real-time system," REPORT-UNIVERSITY OF YORK DEPARTMENT OF COMPUTER SCIENCE YCS, vol. 398, p. 5, 2006.

[6] 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.

[7] "pICASSO Project," Website (German), [retrieved: Sep 2017]. [Online]. Available: https://www.projekt-picasso.de/projekt/

[8] O. Givehchi, J. Imtiaz, H. Trsek, and J. Jasperneite, "Control-as-aservice 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.

[9] "The Internet of Things - Cisco," CloudSim Website, [retrieved: Sep 2017]. [Online]. Available: http://www.cloudbus.org/cloudsim/

[10] 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.

[11] 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.

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

[13] D. Zuehlke, "Smartfactorytowards a factory-of-things," Annual Reviews in Control, vol. 34, no. 1, pp. 129 – 138, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1367578810000143

[14] S. Rostedt, "Intro to Real-Time Linux for Embedded Developers," Linux Foundation Blog, 2013, [retrieved: Sep 2017]. [Online]. Available: https://www.linuxfoundation.org/blog/intro- to- real-time-linux-for-embedded-developers/

[15] J. Huang, "RTMux: A Thin Multiplexer to Provide Hard Realtime Applications for Linux," Embedded Linux Conference Europe, October 2014, [retrieved: Sep 2017]. [Online]. Available: https://events.linuxfoundation.org/sites/events/files/slides/rtmux 1.pdf