Orchestration of Microservices for IoT using Docker and Edge Computing

Muhammad Alam*, João Rufino*, Joaquim Ferreira*, Syed Hassan Ahmed†, Nadir Shah‡, Yuanfang Chen§,

* Instituto de Telecomunicações, Universidade de Aveiro, Portugal.

† Department of Computer Science, Georgia Southern University, Statesboro, GA 30458, USA.

‡ Department of Computer Science COMSATS Institute of Information Technology, Wah Campus, Pakistan.

§ School of Cyberspace, Hangzhou Dianzi University, China

alam@av.it.pt, {joao.rufino, jjcf}@ua.pt, sh.ahmed@ieee.org, {nadirshah82, yuanfang.chen.tina}@gmail.com

Abstract

The world of connected devices has led to the rise of the Internet of things (IoT) paradigm, where applications rely on multiple devices, gathering and sharing data across highly heterogeneous networks. The variety of possible mechanisms, protocols and hardware has become a hindrance in the development of architectures capable of addressing the most common IoT use cases, while abstracting services from the underlying communication subsystem. Moreover, the world is moving towards new strict requirements in terms of timeliness and low latency in combination with ultra-high availability and reliability. Thus, future IoT architectures will also have to support the requirements of these cyber-physical applications. In this regard, edge computing has been presented as one of the most promising solutions, relying on the cooperation of nodes by moving services directly to end-devices and caching information locally. Therefore, in this paper, we propose a modular and scalable architecture based on lightweight virtualization. The provided modularity, combined with the orchestration supplied by Docker simplifies management, enables distributed deployments, creating a highly dynamic system. Moreover, characteristics such as fault-tolerance and system availability are achieved by distributing the application logic across different layer, where failures of devices and micro-services can be masked by this natively redundant architecture, with minimal impact on the overall system performance. Experimental results have validated the implementation of the proposed architecture for on demand services deployment across different architecture layers.

Index Terms

Internet of Things; Microservices; Edge Computing;

I. Introduction

In the last decade, Cloud Computing (CC) has been a trending research topic and received an increasing amount of attention from both the scientific and the industrial community. Ergo, several reputable companies, such as Google, Facebook and Apple, have developed their own cloud-based services which have become a daily work-tool for millions of users. In these systems, data and information are centrally stored and most of the computational workload is performed in the cloud. Such systems are very cost-efficient, scalable, and provide quasi-unlimited data storage for organizations. Yet, with the recent hype surrounding the Internet of Things (IoT) and its applications, both developers and users are experiencing one of the downsides of CC, longer latencies. Where, applications with very strict bounded time requirements, such as smart health, cooperative intelligent transport systems and industrial applications, suffer the most. Furthermore, with the realization that IoT is going to play a vital role in the development of more advanced applications and solutions for the most common problems the production of low-cost smart devices and sensors has reached an unprecedented growth rate. These devices are the driving force behind IoT and produce a tremendous amount of data that, in most cases, must be transformed, processed, stored and analysed on-line. Those computational endeavours are usually centrally handled by cloud-based solutions, but the unique properties of the information received (rate, volume and variety) are saturating the network creating severe management problems.

People rely heavily on Personal Digital Assistant applications, often running on smartphones, tablets and sometimes on wearable equipment (i.e. watches and glasses). As a result, users are increasingly demanding that their personal devices instantly connect and interact with smart objects around them to get information and assistance as well as to use available services without being submitted to complex registration processes. Inside a personal ecosystem, people can be facilitated in many different ways based on their context, preferences and predefined ontologies. These personal edge devices have the capability of generating and processing the vast amount of data that is produced. Therefore, in order to improve the end-user experience, e.g., accessing the requested contents in a shorter time, it would be more reasonable to push some computations closer to these edge devices. Therefore, this idea led to the introduction of a new layer (mediation layer) between cloud and end-devices that has the ability to host small applications and also the Fog Layer [1]. These days, the devices offer more computational power, providing flexibility at the application layer and offering a point of service both to nomadic devices and to central services. The combination of such capabilities with the presented limitations and challenges have paved the way to a new paradigm, Edge Computing (EC). Pushing computing applications, data, and services away from centralized nodes into the logical extremes of

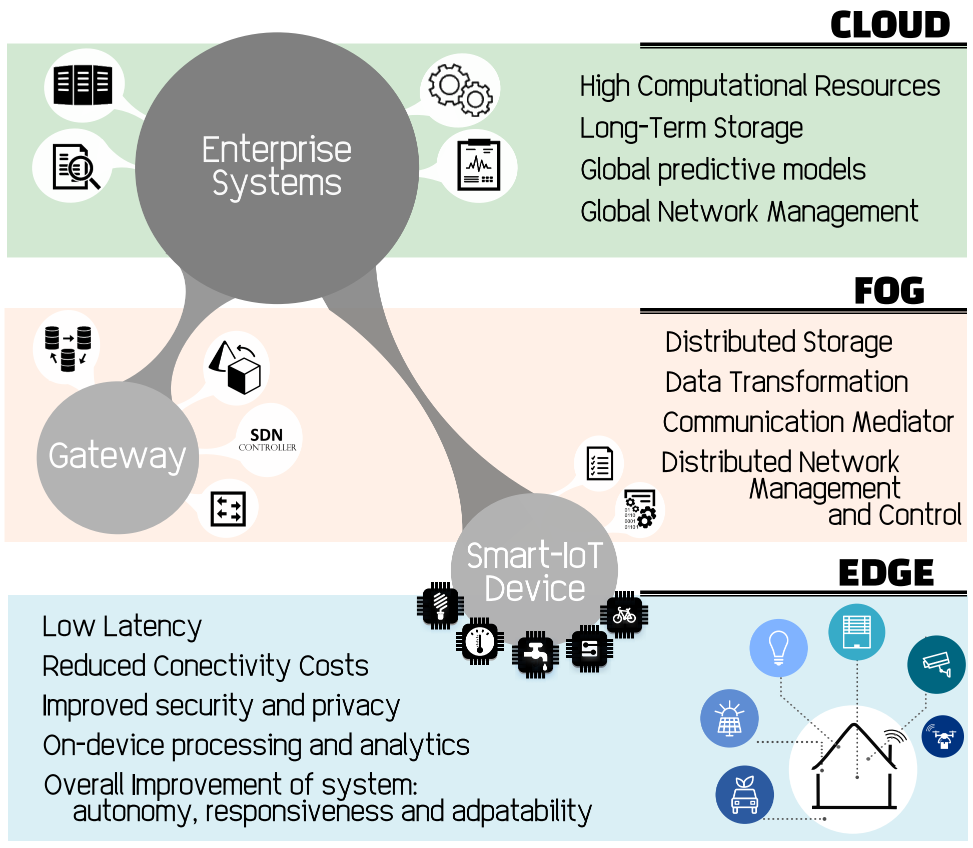
CLOUD
High Computational Resources
Long-Term Storage
Global predictive models
Global Network Management
Enterprise Systems
FOG
Distributed Storage
Data Transformation
Communication Mediator
Distributed Network Management and Control
Gateway
SDN CONTROLLER
Smart-IoT Device
EDGE
Low Latency
Reduced Conectivity Costs
Improved security and privacy
On-device processing and analytics
Overall Improvement of system: autonomy, responsiveness and adpatability

Fig. 1: Representation of Cloud, Fog and Edge Computing.

the network, EC is being presented as the most prominent solution for the aforementioned challenges [2]. This approach requires leveraging resources on connected devices but also on those that may not always be accessible and make them work cooperatively to achieve common goals [3]. As can be seen in figure 2, this on-device approach helps on reducing the latency for critical applications, the cloud-layer dependencies, and better manage the massive data deluge.

In the pursuit of the most suited topology for the presented work in this paper, several architectures were explored in the literature. For instance, in designing a distributed software architecture, the use or consideration of Micro-service [4] is relatively a new concept. This is mainly because the micro-services introduce a way to deal with creating applications as an arrangement of small autonomous services which is completely different from the monolithic architectures design. As a result, various parts of the same application can be independently implemented, making it conceivable to break down huge applications in small services with a particular target to achieve.

On the other hand, achieving visualization through container-based approach is not a new concept; it is done through Dockers [5]. Docker is an open source platform to develop and run distributed applications much faster and independently of the underlying operating systems. Key advantages of using Dockers for distributed applications are less CPU overhead, versioning control, easily portable and improved network performance [6]. Therefore, Docker has been be used in many recent works. For example, in [7], the authors have used Docker as a distributed service platform, as a fault tolerance of services in [8]. For IoT applications, Docker based Gateways were introduced and tested in [9]. Moreover, architectures flexibility and scalability was tested through mirco-services in [10]. It is evident from the previous works that Docker or other container-based technologies have sufficiently penetrated the distributed application development market and could be a reasonable contender for CC, EC and can be used to customize the IoT platform by offering data processing services closer to the end-user. Therefore in this paper, for running the IoT applications at different architecture layers, we proposed an approach of combining Docker and micro-services that runs on the top of a scalable and modular edge computing architecture. In the proposed approach, services isolation is achieved through Containerization while containerized service scalability is achieved through the application of Docker tools. In addition, the applications are treated as independent micro-services that makes the system more modular and decentralized.

The rest of this paper is organized as follows: the proposed architecture and working is presented in section II. A test case scenario that represents the implementation of the proposed architecture using gateways and edge devices is presented in section III. The following section, Section V presents the experimental tests and validation through the obtained results. The last section VI presents major conclusions.

II. Proposed Architecture

In our topology, an application is a set of stateless services that can be distributed across single or multiple machines. In fact, each application is decomposed in independently deployable software components with highly specific functions, encapsulating its implementation. The provided modularity allows the services to be small and loosely coupled and much easier to understand, improving both maintainability and testability. This property is achieved using a containerization software (Docker), embedded in each device, providing the minimum suitable environment for the application.

As depicted in figure 2 three different layers, each growing in computational resources, compose our proposed architecture. The nodes present in the edge layer hold the closest position to the service goal and are capable of sensing, monitoring or communicating with their surrounding environment. Gateways cooperatively cache and process data to reduce the system response time. Therefore, the middle layer mediates the communication between both ends and offers a single entry-point to the Local Systems. Multiple local domains meet at the enterprise layer, where resource-intensive applications run.

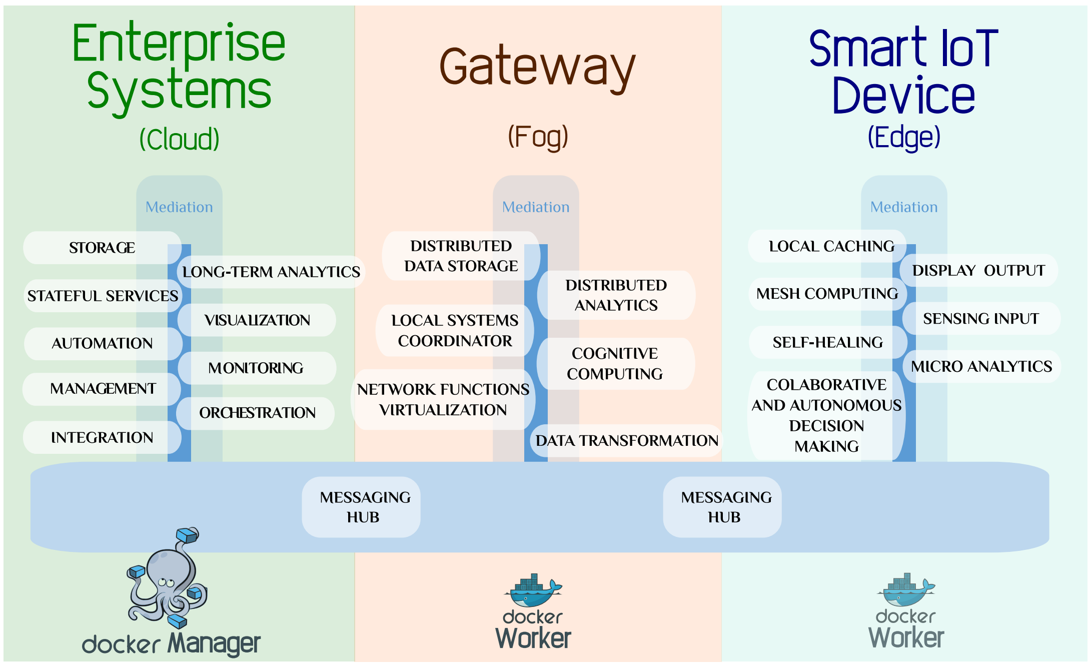

Fig. 2: Overview of the proposed architecture.

In highly dynamic distributed systems, such as the one presented, having an entity capable of managing multiple nodes and their resources is of utmost importance. Hence, the role of the manager is to ensure the most efficient allocation of resources by keeping track of the service deployment. Thus, as an orchestrator, the manager can group, scale and update hundreds or thousands of containers in a cluster of machines. Moreover, the messaging hub was introduced and used to hide the technical specifications and complexities of different components. This multilayer entity allows a manager to define communication channels to applications, reducing the intra-service network overhead.

A. *Smart-IoT Devices*

In our topology, edge devices have an extremely important role as both the human-machine interface and the local environment perceiver. With their computational capabilities duplicating every five years, it is also in this layer where greater innovation is expected. Thus, these devices are presented as cyber-physical systems capable of hosting Docker containers. Using virtualization, we expect to operate over these devices, managing them locally through gateways and orchestrating them centrally in the Cloud.

B. *Gateways*

The middle layer offers a connection point to end layers. In the edge device perspective, gateways offer a local connection point, timely and reduced latency response and coordination support. Moreover, gateways send the status of connected edge devices to the cloud, transform the data received, perform simple analytics and virtualize network functions ensuring users best quality of experience.

In order to achieve adaptability, we followed the software-defined (SD) approach. Following the SD networks procedure, different channels are used for control and data and the cloud can push network rules to gateways that are later enforced by end-devices. Additionally, we use the SD storage strategy and allow storage peers to be deployed in different layers to meet complex requirements. The system can only be fully adaptable if it presents cognitive features. Therefore, data mining is performed in the cloud to define data patterns that can be used by gateways. The SD local intelligence is achieved using the received patterns to adapt accordingly. Its main function is to detect local system erroneous behaviour, check components responsiveness and ensure that edge-devices are complying with system policies.

C. Messaging Hub

Similar to other service-oriented architectures, the key communication component of the proposed is a messaging bus. In our topology, this component is a multi-layer cluster of services capable of handling a huge volume of messages per second. Moreover, each application has a dedicated channel for intra-communication, eliminating direct communication between publisher and subscriber and reducing the subsequent challenges.

D. Enterprise layer

Enterprise systems have three main objectives: data warehousing, treatment and business logistics. Therefore, long-term storage, analysis and management of enormous volumes of unstructured data are conducted at the cloud. Performing this CPU/memory demanding applications at a centralized position allows a granulated model of the overall system, providing predictive capabilities and almost real-time system response to adversity. The resulting models are defined and translated into orchestration rules and disseminated and enforced across gateways, for a supervised control of the local systems.

E. Key Characteristics

In the vision of IoT, all devices are connected in a global network through a plethora of active radio technologies. In addition, the large number of devices that are connected to the IoT produce tremendous amount of data. In a conventional approach (architectures), this data is usually stored and processed at a central cloud greatly reducing and limiting the bandwidth of the communication technologies. Therefore, our proposed architecture makes sure that data gets collected and analyzed at most efficient and logical places between the source and the cloud, balancing the load and pushing the computation and intelligence to the appropriate layers.

The disruptive growth of IoT is continually forcing the industry to answer new challenges. The current monolithic platforms aren't handling the pandemic hardware and software advances. In comparison, while adding new features to such systems would imply a system restructure, in the presented

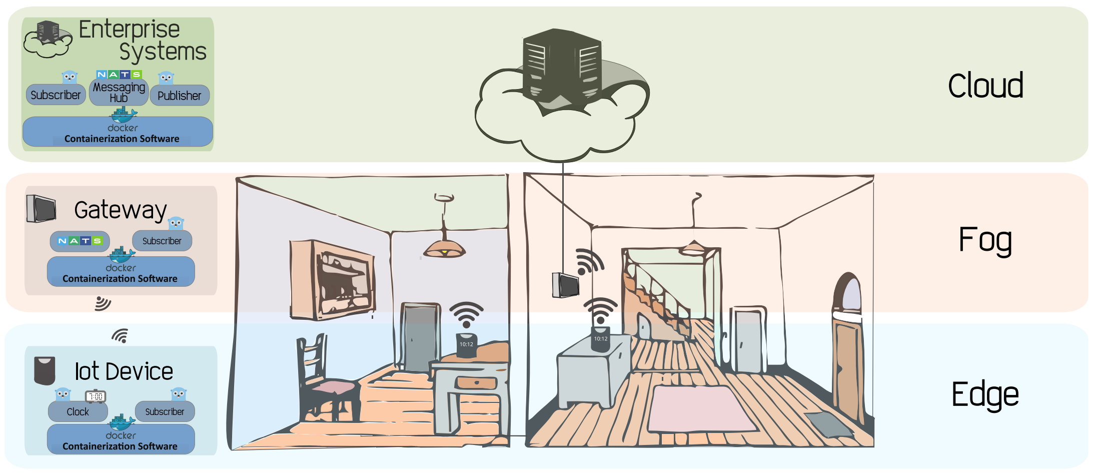

Fig. 3: Test Scenario for multi-layer orchestration.

topology, by combining lightweight virtualization with orchestration we can update the system in real-time. Moreover, the system can spread the application logic across different devices, making use of often under-utilized resources, or even replicate the same service to improve reliability.

In resume, the proposed architecture supports IoTs key attributes: heterogeneity, scalability and adaptability. In fact, in the presented topology, the incorporation of highly modular and loosely-coupled architecture with a messaging hub shields applications from the other-layers technical details and specifications. Consequently, heterogeneous systems can communicate with reduced effort. Scalability is ensured through cachable, replicated, stateless services and a publishing-subscribing system. Furthermore, the orchestrator can adapt the system to meet different requirements guarantying overall evolvability and adaptability.

III. Test Case Scenario

The proposed architecture is validated through a use case scenario that highlights its capabilities for the IoT deployment and is depicted in figure 3. The scenario depicts a typical smart home appliance, two different end-devices are periodically communicating with central servers deployed in the Cloud through a GW. In fact, the GW is being used as a mediator, locally caching the information sent by the ES and distributing it to different edge-devices, alleviating the edge workload and resources used. The end-devices work as cyber-physical systems, that can be remotely controlled and can also interact directly with the user.

In this topology, services can be moved from layer to layer. As stated before, one of the challenge of this architecture is concerned with inter-service dependencies and stateful services. In fact, services with multiple dependencies within different layers reduce the scalability and increase the response time of the system. Therefore, we designed a test case scenario for the deployment of a time-sensitive service. These time-sensitive service is initialized, deployed and scaled by the Enterprise Systems. The service itself is used as proof of concept. It reduces the inter-service dependencies while still sending data to and from the cloud.

IV. Implementation

For the implementation of the test case, easily obtainable and off-the-shelf hardware was used in order to facilitate reproducible results and research. Thus, a total of three Raspberry Pi 3 (RPi) compose both the gateway and end-devices. The Operating System adopted for the Local Systems was Alarm armv7 for Broadcom [11], a minimal operating system for RPi. RPis 3 are small SBCs with 4 ARM Cortex-A53 1.2GHz CPU and 1 GB RAM, while also having integrated WiFi. Composing the edge layer, two RPIs were connected to I2C Display and used to present the data directly to the user. The remaining RPI was used as a gateway forming the Fog Layer. End-devices interact wirelessly (using Wi-Fi) with the GW which is then connected to the internet to interact with the Cloud. The Cloud is comprised of an Intel core i5 laptop computer with 8 GB memory, running Archlinux OS and Docker

For the entire experiment, Docker version 18.05.0-he was the chosen software for creating the cluster topology. Since version 1.12, Docker has an embedded orchestration tool, named Swarm, with two types of roles: workers and managers. A worker can only be used for hosting containers while a manager can also terminate, deploy, update and manage running containers in the cluster. Nodes joining the cluster were labelled according to its position in the architecture, i.e. the gateway as ”FOG”, the end-devices as ”EDGE” and the manager as ”CLOUD”.

The service used for this scenario is publish based. It has three different roles, publishers, that publish data to channels, subscribers, that receive the published data from channels, and a messaging hub, that mediates the communication. The publisher transmits, in each message, the local epoch time in nanoseconds and the message number to a predefined channel. The messaging hub then broadcasts the data to each subscriber. The subscriber receives the data and presents it in the I2c Display (in the console if one isn't available). The behaviour of the entire application is monitored through the transmissions sent to the messaging hub.

In order to reduce the size of the docker images, and subsequently the download and transfer time between layers, we adopted the same programming language used for developing Docker, Golang. The images created are available online in Docker Hub as niplodim/subscriber, niplodim/publisher and niplodim/nats. Both the subscriber and publisher images receive the configuration parameters as environment variables. The last image is a reduced version of the NATS messaging hub, in specific the Graphical User Interface and encrypted channels, were removed.

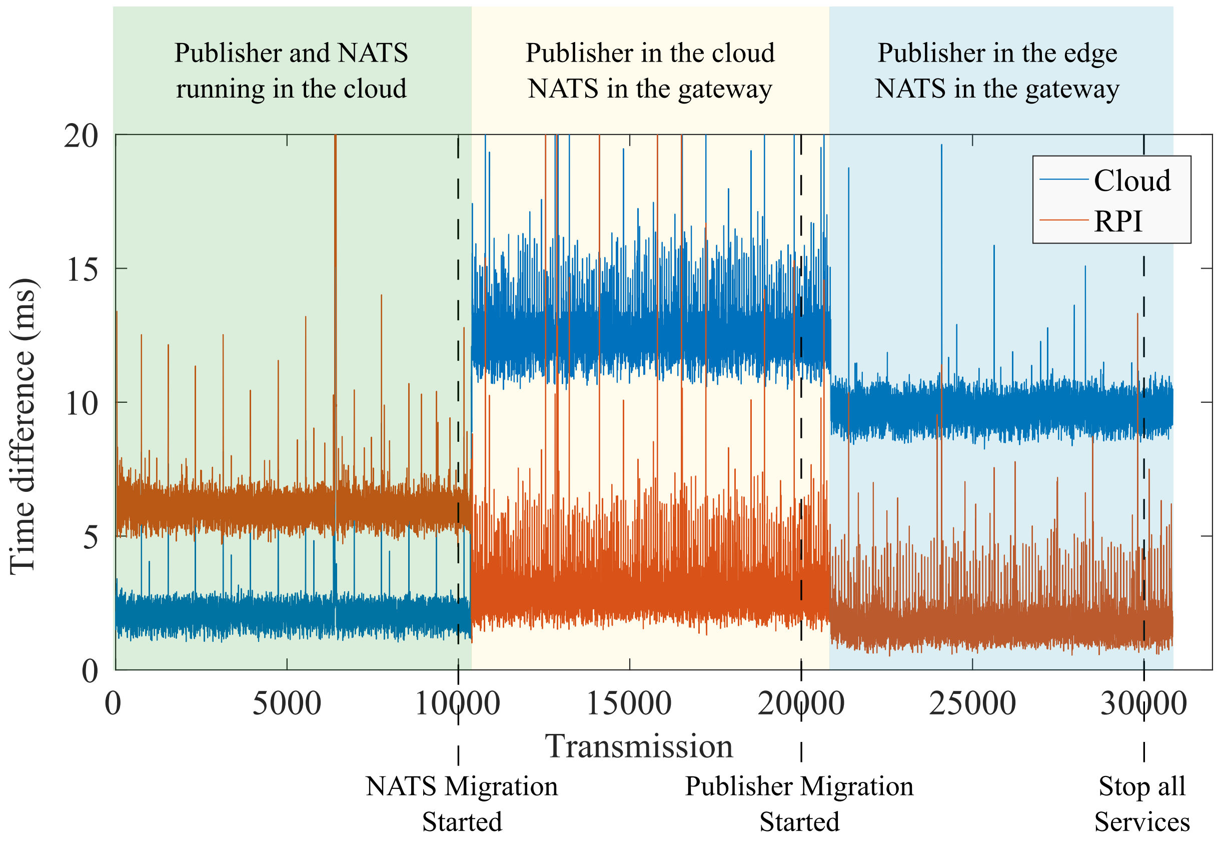

Fig. 4: Communication delay with services in the cloud layer..

V. Results and Discussion

Having a centralized manager allows us to have a grained control over the cloud running services in terms of both orchestration [12] and monitoring [13]. Thus, this topology allows us to provide online service migration from layer to layer without downtime. In order to test this feature, a subscriber, a publisher and a messaging hub were deployed in the cloud and one publisher instance at one end-device (with label EDGE). At the 10000th transmission, the messaging hub is migrated to the gateway and the transmission counter is reset. After other ten thousand transmissions, the publisher is migrated from the cloud to the remaining edge device. The results are depicted in figure 4.

For this experiment, all the docker images used were previously loaded in each node repository. Moreover, ptpd [14], an implementation of the Precision Time Protocol, was used to ensure that the clocks from all nodes were synced. Additionally, it is important to mention that the end to end time comparison is made before the update in the I2C displays. By comparing the different service logs we can see the number of times data has been published and which times were published, and compare them to the time of reception.

Figure 4 presents the end to end service latencies per transmission. With both the publisher and messaging hub running in the cloud, the enterprise systems present the smaller latencies with an

TABLE I: Results summary

	Subscriber Cloud			Subscriber Edge		
Location	**Average** (μs)	**Min** (μs)	**Max** (μs)	**Average** (μs)	**Min** (μs)	**Max** (μs)
Cloud	2052.6	1029.9	8709.9	6198.7	4600.1	74880.0
Fog	11951.8	8668.9	225009.6	2951.4	1155.8	21644.8
Edge	9740.3	8248.3	19628.0	2761.8	515.8	13318.9

average of 2ms in the cloud and 9ms in the Edge I. The cloud messaging hub is then stopped and the gateway starts mediating the communication between edges, this action takes 298ms. With the data being published by the cloud to the gateway the next phase presents the worst results of the experiment in terms of both latency and variance. It is important to notice that the cloud is now subscribing to the same information it is publishing, therefore it has the cost of travelling twice through the internet backbone, with a resulting average of 12ms in the cloud and 3ms at the Edge. The phase ends with the migration of the publisher instance to the empty edge-device which took 537ms to perform. With both publisher and NATS running at the Local Systems the average latency is reduced to 10ms in the cloud and slightly reduces at the edge.

Comparatively, to the average end to end Round Trip Time, measured with the ping Linux command, of 4ms, the application doesn't bring additional burden to the communication. The results show that CC delivers a more stable solution with a smaller variance but, unfortunately, comes short on answering the ultra-low latency required by novel applications, such as augmented and virtual reality and the tactile Internet. The results also show that the orchestration brings no downtime to the service and as little to none overhead to the system. Thus, confirming that the architecture has the ability to aggregate storage resources, scale out the system across different layers and manage the shared resources pool through a central administrative interface.

VI. Conclusions

In this paper, we presented a layered and modular architecture that is running on cloud, fog and edge devices and offer containerized services and micro services. The proposed architecture has three main layers: sensing, mediation and enterprise layer. The sensing layer is the lower layer that performs the sensing and operations (edge computing), the meditation layer represents the intermediate devices and operations (GWs, fog computing), and the upper layer is called the enterprise layer that represents the cloud and operations such as long term global storage. The proposed architecture makes sure that data gets collected and analyzed at most efficient and logical places between the source and the cloud, balancing the load and pushing the computation and intelligence to the appropriate layers. The service can only be accessed through a dedicated channel which allows the services to be small and loosely coupled across the whole system. The proposed architecture ensures higher reliability of the system by using orchestration rules that can achieve full service recovery in and during failures. In addition, the higher system resilience is achieved by introducing redundancy at different layer. The proposed architecture is tested for IoT deployment via smart home use case in which real-time/time sensitive application was tested using publish/subscribe method. The results validate that the enterprise layer has management and control capabilities that ensure application deployment at different layers such as enterprise layer and sensing layer (edge layer). This is mainly achieved by the application of orchestration tools. The orchestration ensures availability of service, the system was up and running and no additional overhead was introduced. In addition, the results also prove that migrating services to the edge-layer can improve the application latency while increasing its variance.

Acknowledgments

This work is funded by National Funds through FCT - Fundação para a Ciência e a Tecnologia, Portugal, under the project UID/EEA/50008/2013.

References

[1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog Computing and Its Role in the Internet of Things," in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ser. MCC '12. New York, NY, USA: ACM, 2012, pp. 13–16. [Online]. Available: http://doi.acm.org/10.1145/2342509.2342513

[2] A. Ahmed and E. Ahmed, "A survey on mobile edge computing," in 2016 10th International Conference on Intelligent Systems and Control (ISCO), Jan 2016, pp. 1–8.

[3] P. Bartolomeu, M. Alam, J. Ferreira, and J. Fonseca, "Survey on low power real-time wireless mac protocols," Journal of Network and Computer Applications, vol. 75, pp. 293–316, 2016.

[4] D. Namiot and M. Sneps-Sneppe, "On iot programming," International Journal of Open Information Technologies, vol. 2, no. 10, 2014.

[5] C. Anderson, "Docker [Software engineering]," IEEE Software, vol. 32, no. 3, pp. 102–c3, 2015. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7093032

[6] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An Updated Performance Comparison of Virtual Machines and Linux Containers," Technology, vol. 25482, pp. 171–172, 2014.

[7] D. Liu and L. Zhao, "The research and implementation of cloud computing platform based on docker," 2014 11th International Computer Conference on Wavelet Actiev Media Technology and Information Processing(ICCWAMTIP), pp. 475–478, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7073453

[8] B. I. Ismail, E. Mostajeran Goortani, M. B. Ab Karim, W. Ming Tat, S. Setapa, J. Y. Luke, and O. Hong Hoe, "Evaluation of Docker as Edge computing platform," ICOS 2015 - 2015 IEEE Conference on Open Systems, pp. 130–135, 2016.

[9] R. Morabito, R. Petrolo, and V. Loscr, "Enabling a lightweight Edge Gateway-as-a-Service for the Internet of Things," pp. 1–5, 2016.

[10] D. Salikhov, K. Khanda, K. Gusmanov, M. Mazzara, and N. Mavridis, "Microservice-based IoT for Smart Buildings," no. Ii, 2016. [Online]. Available: http://arxiv.org/abs/1610.09480

[11] A. L. ARM, "Arch Linux for Arm," https://archlinuxarm.org/, 2018, [Online; accessed 23-May-2018].

[12] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of containerized microservices for iiot using docker," in Industrial Technology (ICIT), 2017 IEEE International Conference on. IEEE, 2017, pp. 1532–1536.

[13] J. Rufino, M. Alam, and J. Ferreira, "Monitoring v2x applications using devops and docker," in Smart Cities Conference (ISC2), 2017 International. IEEE, 2017, pp. 1–5.

[14] "Precision Time Protocol," https://github.com/ptpd/ptpd, 2018, [Online; accessed 23-May-2018].

Biographies

Muhammad Alam (S'10, M'14, SM'17) holds a Ph.D. degree in computer science from the University of Aveiro, Portugal (2013–2014) and M.S. degree in computer science from the International Islamic University Islamabad, Pakistan (2008). He has participated in several EU funded projects such as C2POWER, ICSI, and PEACE. Currently, he is working as an assistant professor at Xi'an Jiaotong-Liverpool University, Suzhou, China. His research interests include IoT, real-time wireless and vehicular communications.

João Rufino is currently pursuing a Master's degree in Mathematics and Applications at the University of Aveiro, Portugal. Since 2016, he has been working as a researcher at Instituto de Telecomunicações - Aveiro, where he is contributing to the creation of a scalable system for the deployment and development of V2X applications. His research interests include scalable systems and vehicular communications.

Joaquim Ferreira received Ph.D. degree in Informatics Engineering from University of Aveiro, Portugal in 2005. Currently, he is an adjunct professor at School of Technology and Management, University of Aveiro and researcher at Telecommunications Institute. He has been involved in several international and national research projects. His research interests include: dependable distributed systems, fault-tolerant real-time communications, wireless vehicular communications.

Syed Hassan Ahmed (S'13, M'17, SM'18) completed his BS from KUST, Pakistan and MS/Ph.D. from Kyungpook National University, South Korea both in Computer Science in 2012 and 2017 respectively. Later, he was a Post-Doc with University of Central Florida, Orlando. Currently, he is a faculty with the CS Department of Georgia Southern University (GSU) at Statesboro, USA, where his research interests include Sensor and Ad hoc Networks, Vehicular Communications and Future Internet.

Nadir Shah is currently an Associate Professor with the COMSATS University Islamabad, Wah Campus, Pakistan. His current research interests include computer networks, distributed systems, and network security. He has authored several research papers in international journals/conferences. He is serving in the editorial board of IEEE Softwarizations, AHWSN and MJCS.

Yuanfang Chen received her Ph.D. and M.S. degrees from Dalian University of Technology, China, and second Ph.D. degree from University Pierre and Marie CURIE, France. She currently works in

Hangzhou Dianzi University as a Professor. She has been invited as the Session Chair of some conferences, the associate editor of Industrial Networks and Intelligent Systems, and the guest editor of MONET journal.