(ITEA 3 – 15017)

open standard APplication Platform
for carS and TrAnsportation vehiCLEs

Deliverable: D 2.3

"Description and Development of Cloud Middleware within the Network"

Work Package: 2

Service Enablers in Intelligent Networks

Task: 2.3

"Network and Cloud Middleware Development"

Document Type:	Deliverable	**Classification:**	Internal
Document Version:	Final	**Contract Start Date:**	01.01.2017
Document Preparation Date:	16.02.2019	**Duration:**	31.12.2019

ITEA 3
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

EUREKA

History

Rev.	Content	Resp. Partner	Date
0.1	Initial document structure	Laaroussi Zakaria	16.02.2019
0.2	Adding new chapter	Laaroussi Zakaria	02.03.2019
0.3	Service provisioning section	Laaroussi Zakaria	06.03.2019
0.4	Adding results and discussion	Laaroussi Zakaria	08.03.2019
0.5	Authorization framework for co-operative C-ITS	Ravidas Sowmya	30.03.2019
0.6	Adding discussion to service provisioning chapter	Laaroussi Zakaria	01.04.2019
0.7	Adding conclusion chapter	Laaroussi Zakaria	03.04.2019
0.8	Adding MEC chapter	Jussi Haapola	05.04.2019
0.9	Adding Bibliography	Laaroussi Zakaria	08.04.2019
1.0	Adding results interpretation to Service provisioning chapter	Roberto Morabito	18.04.2019

Contents

History ii

1 Introduction 1
1.1 Document Structure 1

2 Network and Cloud Middleware Development 2
2.1 Service Provisioning 2
2.2 Cloud based vs. Edge based 3
2.3 Discussion 4

3 Utilising Multi-Access Edge Computing Applications for Automotive Systems 8
3.1 Local Break Out 11
3.2 Host infrastructure setup and prerequisites 12

4 Authorization Frameworks for IoT: Analysis 14
4.1 Analysis 14
4.1.1 Overview 14
4.2 Discussion 16

5 Conclusion 20

List of Figures

2.1 Edge/Cloud service provisioning areas. 3
2.2 Driving area while performing the study 4
2.3 Traceroute command result : Edge/Cloud to In-car OBU. 5
2.4 SoTA delivery: Small software images 6
2.5 SoTA delivery: Big software images 7

3.1 High-level view of Application Life Cycle Management. 9
3.2 Application Life Cycle Management operations at packaging and operational levels. 10
3.3 Local Break Out Interfaces. 11
3.4 Application Life Cycle Management virtual machine procedures for setup. . . . 13
3.5 Host machine precedures for setup. 13

4.1 Analysis of existing authorization frameworks for IoT with respect to the evaluation criteria concerning the access control system 15
4.2 Analysis of existing authorization frameworks for IoT. Symbol † indicates that only a toy example is provided for demonstration purposes, but the framework is not designed specifically for that application domain. 16

List of Tables

1 Introduction

The emerging vehicular applications require computing capacity and network connectivity availability. Mobile Edge Computing(MEC) as new 5G feature which focuses on moving computing resources to the edge of networks, complements cloud computing by solving the latency constraints and reducing ingress traffic to the cloud. In this document, our main goal is to investigate the added value by edge from the power of computing.

1.1 Document Structure

This deliverable is organized as follows: In chapter 2 we described the continuation of the testbed selected for the SoTA use-case defined in D2.2, in chapter 3 we introduced the Multi-access Edge Computing (MEC) applications and services environment, in chapter 4 we described the components of a novel authorization framework dedicated for Intelligent transportation system. Finally, in chapter 5 we give a summary about the research studies and experiments presented in the document.

2 Network and Cloud Middleware Development

2.1 Service Provisioning

An efficient service provisioning can be very important to achieve the 5th level of autonomous vehicles. The multiple services defined by ETSI for Intelligent Transportation System(ITS) will enable the vehicle to travel from an departure to arrival without a human interaction. In the meantime service providers are aiming to offer services that follow vehicles and drivers everywhere.
In this regard, Cloud providers aim to deliver a better quality of services no matter the throughput offered by the network and the latency that carries caused by different factors.
The 5G mobile network offers new technologies to ensure performing service delivery using different features such as Network slicing, Mobile Edge computing. The contribution reported in this deliverable include the results achieved through the testbed already presented in the deliverable D2.2 . Our contribution to this deliverable is about exploring the possibility of placing the service provisioning at the network edge, by exploiting the Multi Access Edge Computing (MEC) concept in vehicular scenarios and investigate if it can be complementary to what the Vehicular Cloud Computing is already capable of delivering.
In this section, we empirically show–through the setup of a vehicular system testbed—whether a service provisioning issued at the network edge can generate performance benefits in comparison to service provisioning from the centralized cloud. We focus on vehicle communications enabled by enhanced mobile network, by also empirically investigate how the SoTA is affected when it is provided by edge or cloud facilities, while varying the dimension of the software image size..
The results are presented and well presented and discussed in the next sections.

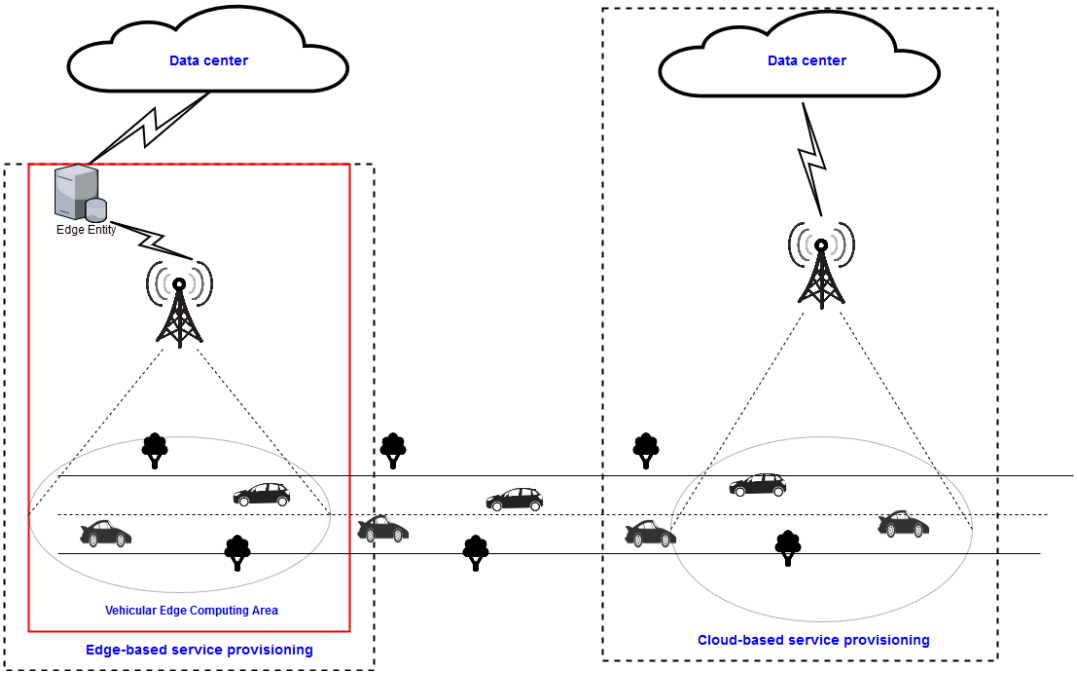

Figure 2.1: Edge/Cloud service provisioning areas.

2.2 Cloud based vs. Edge based

In the past decade, vehicles have grown from just being mechanical to turn connected and smart. Today's vehicles embed enhanced communication abilities with different entities relying on Vehicle-to-Everything(V2X) protocols. In this study we focused on one specific protocol Vehicle-to-Cloud(V2C)/Vehicle-to-Edge(V2E), we evaluated one ETSI service defined for ITS dubbed Software update over the air(SoTA).
In the analysis shown in D2.2, we evaluated and compared different application layer protocols in vehicular scenarios while transmitting small sized workloads. The purpose of the empirical analysis presented in D2.2 was assessing the performance in terms of throughput and latency of different application layer protocols, in the context of vehicular scenarios.
In this new analysis, we focus on evaluating cloud vs. edge-based over the air software update we took the download time as a metric while varying the software image size.
The used vehicular system architecture to evaluate cloud vs. edge-based over the air software update is shown in Fig.2.1, while the main purpose of the first study of the testbed was evaluating the application layer protocols in vehicular scenarios.
In this study, the primary objective was assessing the impact of software images size on the performance in terms of download time in vehicular scenarios. From these results, it also becomes possible to rank the performance variation between edge-based provisioning and cloud-based provisioning.
Fig.2.1 shows two service provisioning areas: Cloud-based area"Ericsson research datacenter located in **Sweden**" and Edge-based area "Edge entity located in Ericsson **Finland**". As a remark, the used edge entity is not following the Mobile Edge Computing(MEC) Specification defined by ETSI, the used server is a workstation running an HTTP server "NGINX," all the technical details of the used implementations and the hardware specifications are listed in D2.2.

Fig.2.2 shows the route of the vehicle while performing the study by receiving a variation of new software images from the cloud and edge respectively.

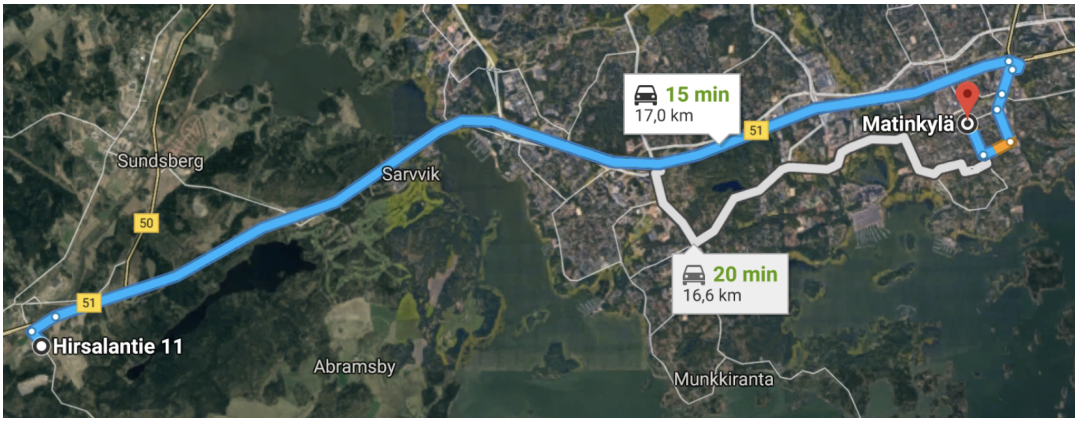

Figure 2.2: Driving area while performing the study

2.3 Discussion

In this section, we present the results of our study; besides, we explain on the empirical outcomes achieved during our performing the testbed.

File sizes ranges are from 100 kB to 1GB; we divided the results to two figures Fig.2.4 and Fig.2.5 in purpose to show the variation we got between small size and big size software images. However, it is worth mentioning that the vehicle's speed variation it wasn't taken into account, the average speed was 80 km/h in Highway road.
The main insights that can be derived from this analysis that there is a dependency between the performance of a cloud/edge and the file size as showing in Fig.2.4 and Fig.2.5. It appears how the edge-based service provisioning is less performing comparing to cloud-based service provisioning in small size software images case, and better performing in the big software images case.

This difference in percentage between edge and cloud depends on to the number of routers that a packet passes through from the in-car OBU to the destination(Edge/Cloud), we used the traceroute command in the aim to trace the route of the packet from a departure to a destination as showing in Fig.2.3, The upper part of the picture shows the results of the traceroute command applied when the provider of the new software image is the data center, and the lower part of the picture shows the results of the traceroute command applied when the provider of the new software image is the edge entity. The main insights that can be derived from the picture are the number of hops to the edge entity is much higher than the one from the cloud, even the vehicle is closer to the edge than the cloud.
Before examining the achieved results, it is worth clarifying what is the existing network setup in the testbed as it could help on understanding the rationale behind the empirical outcome. Both cloud and edge server are interfacing with the base station through the mobile network of a Finnish operator. This also implies that there is no local breakout between the base station

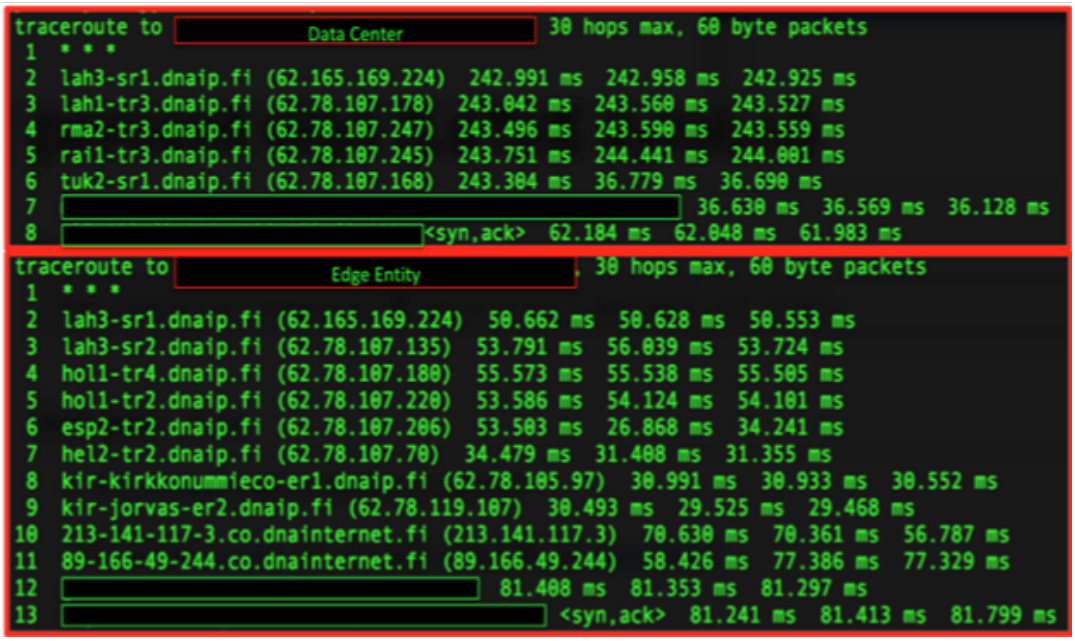

Figure 2.3: Traceroute command result : Edge/Cloud to In-car OBU.

and the edge server–as shown from the traceroute analysis–fully relying on the network setup provided by the mobile network operator

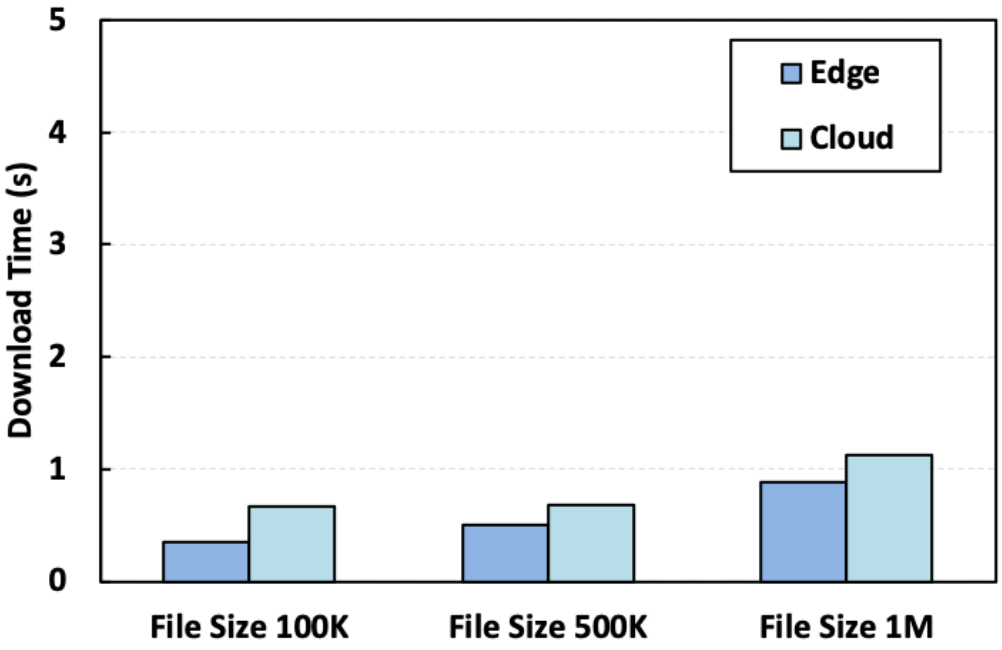

Figure 2.4: SoTA delivery: Small software images

As you can see from Fig.2.4, we are reporting the download time of three size files (100Kb, 500Kb, 1Mb), for both providers Cloud and Edge.
If we look at the graph, we notice that the edge is performing better than the cloud in the case of small size software images.

In Fig.2.5, we are reporting the download time of three size files (200Mb, 500Mb, 1Go) for both Cloud and Edge providers .
The figure shows the performance of both edge and cloud from a download time perspective. As it appears the cloud is performing better than the edge in the case of big software images size.
The achieved results provide very interesting insights. In fact, it is somehow surprising how the network setup of the mobile operator can affect the performance of the service provisioning. In other words, we would expect that the edge-based provisioning is always outperforming the cloud-based provisioning because of the closer physical location between edge server and car. However, this kind of outcome is verified only in the case of Software-over-the-air (SOTA) of small size files. Differently, for the case of SOTA of large size files cloud-based provisioning and outperforming the edge-based one. Although further research investigation is needed for giving an empirical reasoning to these results, it is reasonable to assume that the higher number of hops in the edge-based provisioning produce a longer download time as a result of the higher processing time that each hop must to do of large size files, despite the lower latency between the different hops. Differently, this kind of result does not appear visible in the case of small size files transmission because the need of processing small size files is less resource-demanding and the lower latency between hops in the edge case makes the download time lower–while in the cloud case the longer latency between the different hops is predominant respect the time needed to process the single file.

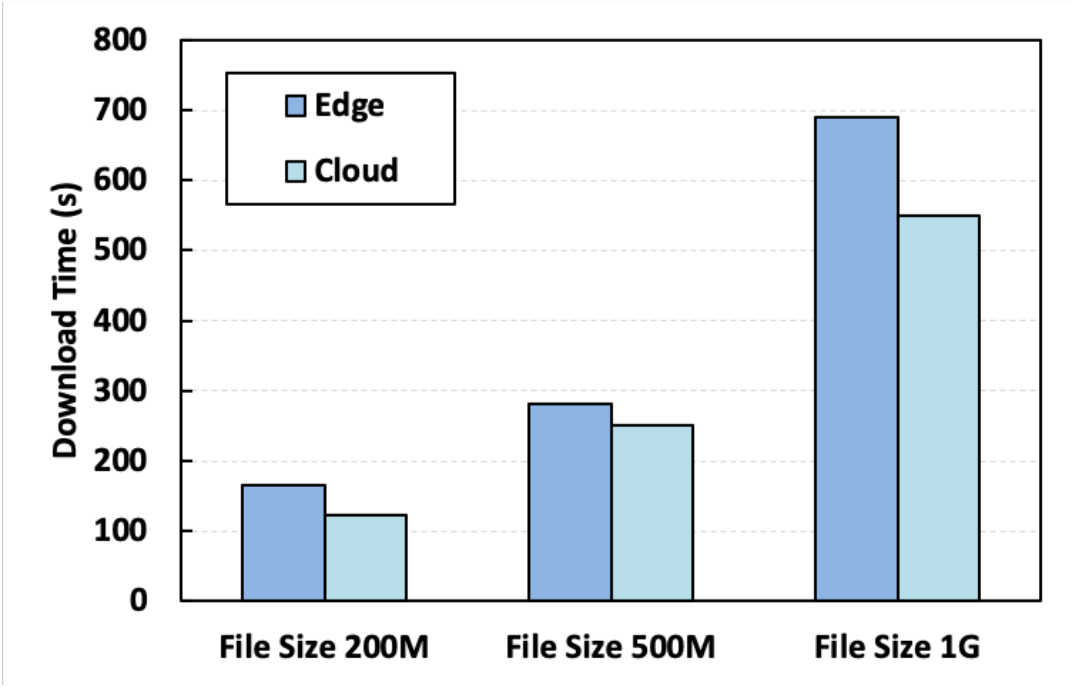

Figure 2.5: SoTA delivery: Big software images

In the following sections, we take a more detailed look on instantiating an automotive application at the edge entity MEC server. The edge entity is ETSI MEC specification compliant solution where the aim is to directly capture the user data plane traffic at the eNodeB. As compared to the illustration in Fig.2.1 the edge entity has a wired connection to the mobile base station. By capturing the LTE user plane traffic, all the additional hops from Fig.2.3 can be eliminated, only the locally applied switching applies. The next sections describe in detail the procedures required for MEC application installation and operation.

3 Utilising Multi-Access Edge Computing Applications for Automotive Systems

Multi-access Edge Computing (MEC) applications and services are installed on virtualised x86 based Commercial Off The Shelf (COTS) Information Technology (IT) Hardware (HW) using the MEC service named Application Life Cycle Management (ALCM). The ALCM is a software-only solution, provided as MEC service, to manage the application life cycle on virtualised x86 based COTS IT hardware. It's high-level view is depicted in Figure 3.1. The ALCM provides Structured Command Line Interface (SCLI) to manage the life cycle operations of applications.

Multiple instances of application virtual machines (VMs) can be deployed and managed by ALCM VM on the same COTS IT hardware host. ALCM and the applications which are managed by ALCM is packaged in Topology and Orchestration Specification for Cloud Applications (TOSCA) Cloud Service Archive (CSAR) format.

The Topology and Orchestration Specification for Cloud Applications is a standard developed under the OASIS foundation. It covers the definition of complex enterprise applications which consists the different loosely coupled components. Components can be tied by using requirements and capabilities which are part of TOSCA standard. One standard for defining TOSCA-based application packages is Cloud Service Archive.

The ALCM validates an application package and creates incubation. This helps to install and launch application VMs by creating necessary resources such as virtual central processing units (vCPUs), networks, memory, etc. The ALCM is the only VM, which interacts directly with the host through the host-only network, with limited privileges. The other applications can be instantiated and managed using ALCM. The ALCM uses the Operation, Administration, Maintenance (OAM) network to interact with the other applications during operations like post-configuration and upgrade.

The functionalities of ALCM are listed as follows:

1. Transfer the TOSCA CSAR of the application to the ALCM VM.
2. The ALCM VM validates the application package against the TOSCA standards.
3. The ALCM VM provides the basic environment for all the application VMs in terms of resource and infrastructure like volumes, memory, and network resources to instantiate.
4. The ALCM VM uses the Ansible framework to do post-configuration operations of the application. The application defines the required tasks to be done as part of post configuration using Ansible playbook which is part of application package.
5. The ALCM VM provides upgrade support for all the applications.

Ansible is an open-source software provisioning, configuration management, and application deployment tool. It runs on many Unix-like systems, and can configure both Unix-like systems as well as Microsoft Windows. It includes its own declarative language to describe system

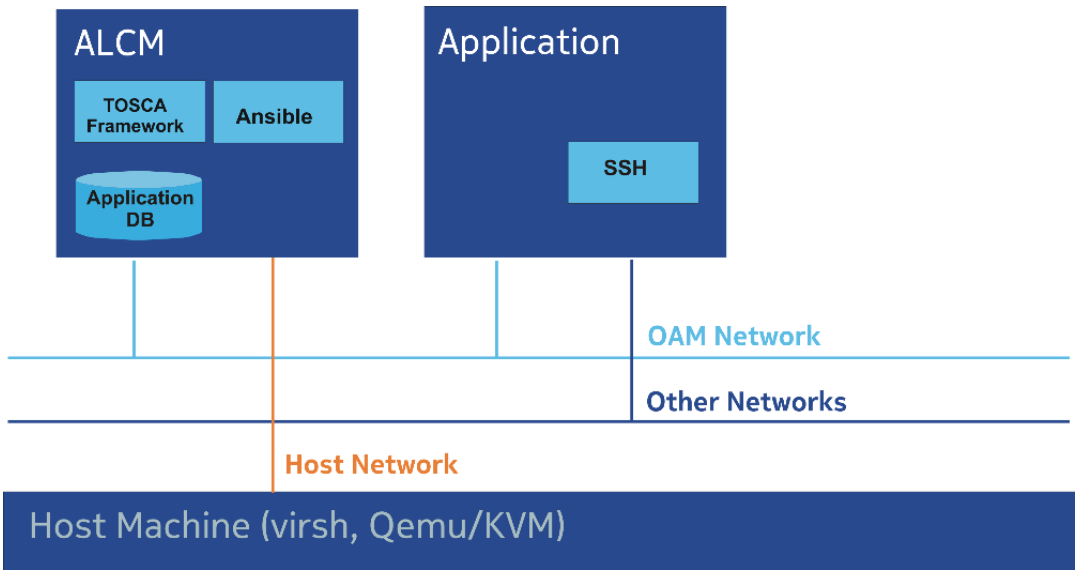

Figure 3.1: High-level view of Application Life Cycle Management.

configuration. Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished.

The ALCM allows following life cycle operations to manage the application package and the application.

As depicted in Figure 3.2 (left side) there are four life cycle operations at an application package level:

- On-boarding: During this operation, the application package will be copied to a pre-defined location inside ALCM VM. The package will be extracted by ALCM and verified.
- Enabling: The application package will be used for instance level operations.
- Disabling: The application package cannot be instantiated. However, an already instantiated instance can be kept in service.
- De-boarding: Any resource created and stored package will be deleted from ALCM VM. A package can be de-boarded only when no application instance of that package exists.

There are also four life cycle operations at an application operation level, as depicted in Figure 3.2 (right side) :

- Instantiating: ALCM prepares the computing, network, and storage resources required by the application. Once the resources are available, the application is booted up.
- Starting: ALCM starts a stopped application
- Stopping: ALCM stops an instantiated or started application
- Terminating: ALCM removes all resources allocated to the application.

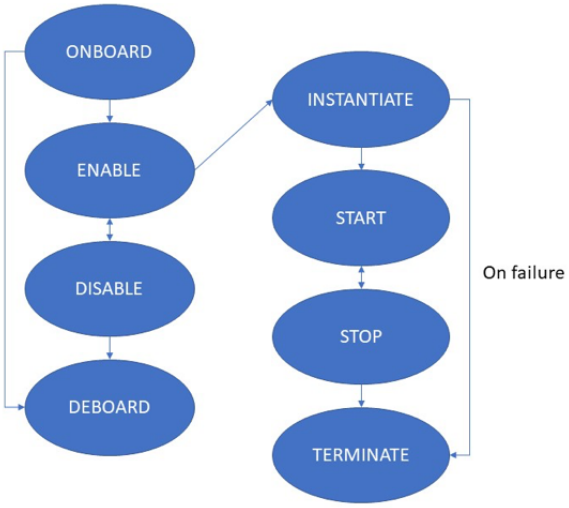

Figure 3.2: Application Life Cycle Management operations at packaging and operational levels.

A terminating application is configured usually with a unique IP address across the entire operator network. This allows using the IP address directly in the client. The IP address of the OAM network for the application is a static OAM address provided by an operator. In case an application requires licenses to work, the complete licensing should be managed by the application itself.

Applications that are managed by ALCM are mandated to describe their application in TOSCA format. The application image and all the necessary files along with TOSCA YAML (YAML Ain't Markup Language), needs to be verified for their authenticity to ensure they are not tampered by any unknown entities. YAML is a human-readable data serialisation language. It is commonly used for configuration files, but can be used in many applications where data is being stored or transmitted. YAML targets many of the same communications applications as XML but has a minimal syntax which intentionally breaks compatibility with standard generalised markup language. The ALCM provides a framework to authenticate and to authorise the package before deployment. This is done via the Certificate Authority (CA) who signs the application package. If some specific performance requirements are needed, application developer can configure non-uniform memory access (NUMA) nodes to allocate desired CPUs and memory for the application.

Loading Run Time Environment Settings

Run Time Execution (RTE) for applications can be loaded by mounting the CDROM image as part of application boot sequence. The RTE entities can be variables or scripts that is used for application startup.

Accessing Run Time Execution using config_drive

An application developer can use the 'config_drive' inside the CSAR package to define what configuration should be made available by the operator. This is an optional service that can be

utilised when the application is deployed using MEC. The 'config_drive' contains "user_data" file, which has information of network interfaces, storage volumes, etc. With Network Interface Info, an application can configure IP, route, and so forth. With storage volumes information, an application can mount the file systems. The application developer should decide what to do with the information in 'config_drive'.

The application developer can make use of "cloud-init" package to read the "user-data". To make the "cloud-init" work, the application developer should configure it accordingly. The MEC instantiation recommends using the "cloud-init" package.

Configuring RTE via files

An application developer may optionally choose a set of files that need to be present when the application virtualised network functions (VNF) boots up. These files are passed transparently from the MEC to the application at the deployment. These files must be present in the CSAR package and should be appropriately linked in the TOSCA YAML. Along with the "user-data" file, these files will be available in the CDROM image.

3.1 Local Break Out

The virtual Serving Gateway – Local Break Out (vSGW-LBO) VNF provides a break-out functionality for LTE traffic based on operator configured traffic offload policies. In APPSTACLE domain this functionality would include, for example, the emergency warnings scenario, where notification of a detected road hazard would be managed by a nearby MEC entity, or some other scenario needing to manage locally dense deployments (e.g. major intersections), or very low latency solutions. The Local Break Out (LBO) functionality is deployed and distributed at the edge of the network. The LBO is interoperable with the operator's Mobility Management Entity (MME) and Physical Gateway (PGW) through the 3GPP standard interfaces S11 and S5/S8, respectively. These VNF are 3GPP compliant, where the SGW is a part of virtualised MEC (vMEC) platform controlled and coordinated from the central core. It allows the traffic breakout towards special application servers located in the vMEC platform. The vSGW-LBO selection is based on the International Mobile Subscriber Identity (IMSI) number.

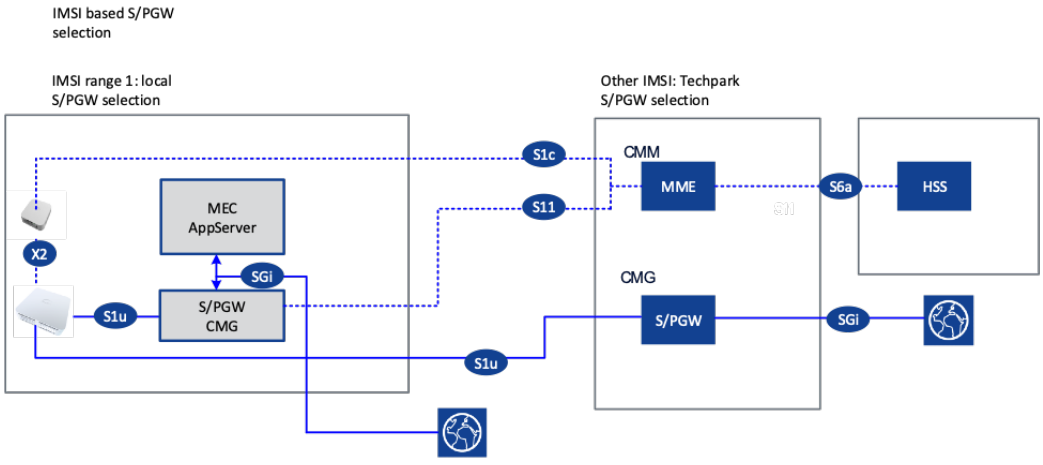

Figure 3.3: Local Break Out Interfaces.

Figure 3.3 depicts the interfaces of the LBO. Here, the X2 is the LTE control plane external interface defined between two neighbour eNodeBs. The S1 interface in LTE is used between eNodeBs and the EPC; specifically S11 between the MME and SGW. S1u is the S1 user plane external interface, S1c is the S1 control plane interface, sGi is the interface defined between the PGW and external networks, and S6a is the interface is between the MME and Home Subscriber Server.

3.2 Host infrastructure setup and prerequisites

Any instantiation of the MEC VNF has a number of prerequisites. On any server that will host MEC VNF, the following BIOS settings are mandatory or strongly recommended.

- Intel VT-x (Intel Virtualization Technology for IA-32 and Intel 64 Processor) is enabled.
- Intel VT-d (Intel Virtualization Technology for Directed I/O) is enabled.
- Intel IOMMU is enabled in the host to support SRIOV based NIC cards.
- Apparmor is enabled (Ubuntu).
- Hyper-threading is enabled.
- Ubuntu 18.04 is used as an operating system.

On a Linux environment the following required packages must be installed: “QEMU”, “KVM”, “virt-manager”, “bridge utils”, and “libglu” (qemu-img, qemu-kvm, libvirt, bridge-utils, genisoimage). Basic requirements for hardware is to have x86 family processor with at least 2x12 cores. 2x 800GB of SSD disks and Intel SRIOV and/or DPDK compatible NICs with 1Gb OAM port for host and 2x 10GB port for vMEC. Virtual Functions (VF) are created and attached to network device (NIC). A MEC user with administrator privileges is created to host ALCM. Lastly, Figures 3.4 and 3.5 procedures are carried out by the ALCM VM and host machine, respectively.

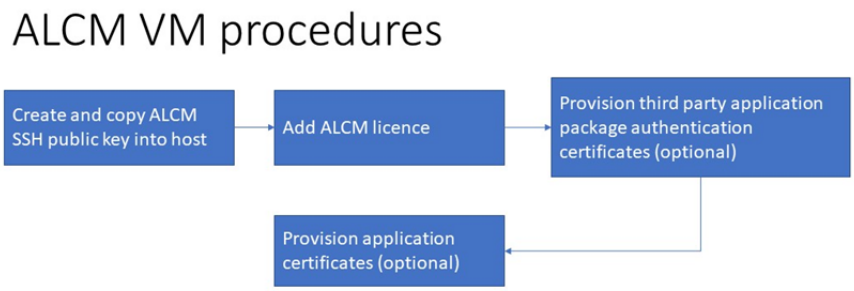

Figure 3.4: Application Life Cycle Management virtual machine procedures for setup.

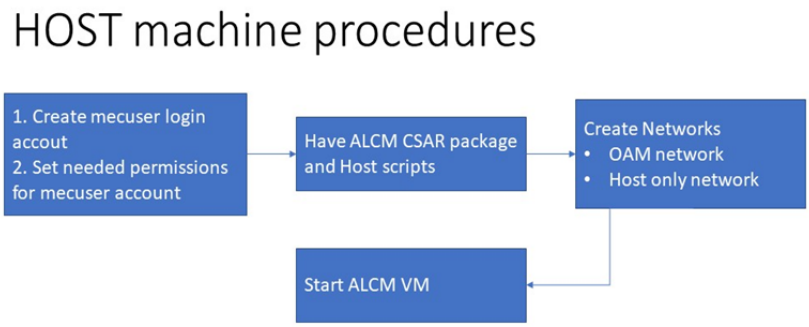

Figure 3.5: Host machine precedures for setup.

4 Authorization Frameworks for IoT: Analysis

In deliverable D2.1, we discussed the generic IoT requirements and the access control requirements derived from it. We also elaborated on different architecture styles; namely, the centralized, connected and distributed, and discussed the state of the art research. In this section, we discuss typical middlewares that is used in IoT eliciting their benefits. Furthermore, we provide an analysis of existing authorization frameworks for IoT based on criteria such as architecture style and where the authorization components are deployed (in physical node, edge or cloud).

Middleware The middleware is often required to ensure connectivity, interoperability, storage and computation of data within an IoT ecosystem. Different types of middleware have been proposed for IoT [31, 37, 38]. In our study of the literature on access control for IoT, we observed that most of the existing solutions rely on cloud computing and edge computing as middleware. According to the NIST [25], *cloud computing* is a computing paradigm that enables on-demand network, storage, applications and other services without any management effort. Cloud computing has become a core component of most of IoT platforms because it provides elastic and scalable data storage and processing. The adoption of cloud computing has opened new directions for technological enhancements in several IoT applications. The cloud has different role within the IoT architecture, depending on the application needs and requirements.

Fog computing [39] and *edge computing* [11] have been recently proposed as new computing paradigms to reduce the communication latency and bandwidth required by the use of a remote cloud platform for data storage and processing. The underlying idea behind fog computing is that data generated by IoT devices are processed at the edge of the network, close to where they are generated. A network of micro data centers process and store critical data locally and then push all received data from IoT devices to a remote cloud platform for long-term storage. The computational processes being done by the micro data centers is usually referred as edge computing, while the network connections between the micro data centers and the cloud platform is referred to as fog computing.

4.1 Analysis

4.1.1 Overview

Our analysis of the literature shows that a variety of approaches have been designed and developed to enable access control in IoT. These approaches can be broadly classified in two main categories based on the policy evaluation strategy and architecture. On one side, we have authorization frameworks [1, 2, 3, 4, 7, 8, 9, 12, 13, 18, 19, 20, 22, 23, 26, 28, 29, 30, 33, 35, 40, 41, 42] that adopt a policy-based architecture and a runtime policy evaluation strategy. Most of these frameworks are inspired to the XACML standard. On the other side, we have

frameworks [6, 14, 15, 16, 17, 23, 32, 36] that adopt a hybrid-based architecture and policy evaluation strategy. A number of these frameworks build on top of OAuth by extending this standard to enable the generation of tokens based on the evaluation of access control policies like in [6, 9] whereas Rivera et al. [32] adopt UMA.

Regardless the type of access control architecture, different deployments and technologies are used to implement architecture. For example, PDP, PEP, PAP, and PIP could all be deployed in the cloud like in [1, 2, 26] or they could all be implemented on edge devices [19, 40] or a combination of both [33]. Some works [7, 8, 28] have also proposed authorization mechanisms based on blockchain technology.

Existing frameworks also vary significantly for maturity level. While a few [4, 6, 10, 12, 15, 16, 18, 19, 22, 24, 23, 26, 36] provide a prototype implementation, many [1, 2, 3, 7, 8, 9, 13, 14, 17, 20, 28, 29, 30, 32, 33, 35, 41] only remain at a conceptual level. In particular, Ray et al. [30] and Zhang and Tian [42] only propose an access control model tailored to IoT ecosystems and do not provide detail of the underlying IoT architecture and access control mechanism.

	Evaluation Strategy	Architecture Style	Deployment			
			PAP	PDP	PEP	PIP
Neisse et al. [78]	run-time	policy-based	cloud	cloud	cloud	cloud
Alshehri et al. [22]	run-time	policy-based	cloud	cloud	cloud	cloud
Alshehri et al. [23]	run-time	policy-based	cloud	cloud	cloud	cloud
Fremantle et al. [44]	off-line	token-based	–	external service	cloud	–
Fernandez et al. [41]	run-time	policy-based	external service	external service	local service	–
Cirani et al. [32]	hybrid[f]	hybrid	external service	external service	local service	–
Rivera et al. [89]	off-line	hybrid	external service	external service	local service	–
Seitz et al. [98]	hybrid[c]	hybrid	cloud	cloud + physical node	physical node	physical node
Salonikias et al. [93]	run-time	policy-based	cloud	edge	edge	cloud
Ye et al. [118]	run-time	policy-based	physical node	physical node	physical node	physical node
Hussein et al. [57]	hybrid[c]	hybrid	external service	external service + edge	edge	–
Hernandez-Ramos et al. [55]	hybrid[c]	hybrid	external service	external service + physical node	physical node	physical node
Gusmeroli et al. [52]	hybrid[f]	hybrid	external service	external service	physical node	–
Garcia et al. [46]	run-time	policy-based	physical node	physical node	physical node	physical node
Dorri et al. [37, 38]	run-time	policy-based	edge	edge	edge	–
Ouaddah et al. [81]	run-time	policy-based	physical node	edge + physical node	physical node	physical node
Kim et al. [66]	run-time	policy-based	edge	edge	edge	external service
Tian et al. [108]	run-time	policy-based	edge	edge	edge	edge
Zhang & Tian [121]	run-time	policy-based	?	?	?	?
Jindou et al. [60]	run-time	policy-based	external service 1	external service 1	physical node	external service 2
Guoping & Wentao [51]	run-time	policy-based	external service 1	external service 1	external service 1	external service 2
Islam et al. [58]	hybrid[c]	hybrid	cloud	cloud (+ physical node)	cloud (physical node)	cloud
Barka et al. [25]	run-time	policy-based	external service 1	external service 1	external service 2	–
Cirani & Picone [31]	hybrid[f]	token-based	external service	external service	local service	–
Bouij-Pasquier et al. [28]	run-time	policy-based	external service	edge	edge	physical node
Lee et al. [68]	run-time	policy-based	external service	external service	external service	–
Ray et al. [87]	run-time	policy-based	?	?	?	?
Mahalle et al. [69]	run-time	hybrid	physical node	physical node	physical node	–
Pinno et al. [85]	run-time	policy-based	physical node	physical node	physical node	physical node + external service
Kim et al. [64]	run-time	policy-based	edge	edge	edge	edge
Sciancalepore et al. [97]	run-time	policy-based	physical node	application node + physical node	physical node	external service
Mahalle et al. [70]	run-time	policy-based	physical nodes	physical node	physical node	peers
Schuster et al. [96]	run-time	policy-based	?	?	?	physical node + external service

Figure 4.1: Analysis of existing authorization frameworks for IoT with respect to the evaluation criteria concerning the access control system

	IoT Architecture style	Communication Protocol	Data Format	Cross-domain Data Semantics	Node Failure Robustness	Application Domain	Maturity Level
Neisse et al. [78]	centralized	MQTT	?	No	Yes	–	prototype
Alshehri et al. [22]	centralized	–	–	No	Yes	Lighting†	design
Alshehri et al. [23]	centralized	–	–	No	Yes	Connected Vehicles†	design
Fremantle et al. [44]	centralized	MQTT	JSON	No	No	–	prototype
Fernandez et al. [41]	centralized	–	–	No	No	–	design
Cirani et al. [32]	distributed	6LoWPAN CoAP	?	No	No	–	prototype
Rivera et al. [89]	centralized	–	–	No	No	Traffic Lights†	design
Seitz et al. [98]	distributed	CoAP	XACML JSON	No	Yes	–	prototype
Salonikias et al. [93]	connected	–	–	No	Yes	Connected Vehicles	design
Ye et al. [118]	distributed	–	–	No	No	Wireless Sensor Network†	design
Hussein et al. [57]	connected	?	JSON	No	No	Smart Home†	prototype
Hernandez-Ramos et al. [55]	connected	6LoWPAN CoAP	JSON	No	Yes	–	prototype
Gusmeroli et al. [52]	connected	–	–	No	Yes	–	design
Garcia et al. [46]	distributed	6LoWPAN	?	No	No	Medical Sensor Networks	prototype
Dorri et al. [37, 38]	distributed	–	–	No	Yes	Smart Home†	design
Ouaddah et al. [81]	distributed	–	–	No	No	–	design
Kim et al. [66]	connected	–	–	No	No	Smart Home	design
Tian et al. [108]	connected	HTTP	?	No	Yes	Smart Home	product
Zhang & Tian [121]	?	–	–	No	No	–	design
Jindou et al. [60]	connected	HTTP	JSON	No	No	Smart Home†	prototype
Guoping & Wentao [51]	centralized	–	–	No	No	–	design
Islam et al. [58]	connected	–	JSON	No	Yes	Health Prescription Assistant	design
Barka et al. [25]	centralized	–	–	No	No	–	design
Cirani & Picone [31]	distributed	CoAP	?	No	No	–	prototype
Bouij-Pasquier et al. [28]	connected	CoAP	JSON	No	No	Health IoT†	prototype
Lee et al. [68]	distributed	Wi-Fi	?	No	No	Smart Lock†	prototype
Ray et al. [87]	?	–	–	No	No	Remote Healthcare Monitoring	design
Mahalle et al. [69]	distributed	Wi-Fi	?	No	No	–	prototype
Pinno et al. [85]	distributed	–	–	No	Yes	–	design
Kim et al. [64]	centralized	ZigBee	?	No	No	Smart Home	prototype
Sciancalepore et al. [97]	connected	–	–	No	No	–	design
Mahalle et al. [70]	distributed	?	?	No	No	–	prototype
Schuster et al. [96]	distributed	HTTP	?	No	Yes	Smart Home	prototype

Figure 4.2: Analysis of existing authorization frameworks for IoT. Symbol † indicates that only a toy example is provided for demonstration purposes, but the framework is not designed specifically for that application domain.

4.2 Discussion

The ability to automate the evaluation of an access request is an important requirement in all IoT applications where a multitude of devices and users share information. Assuming that a user is always available to evaluate if access to certain resource should be granted is not realistic. To automate the evaluation of an access request, existing frameworks adopt either a policy-based [1, 2, 3, 4, 7, 8, 9, 12, 19, 20, 22, 23, 26, 28, 29, 30, 33, 35, 40, 41, 42] or a hybrid architecture [6, 14, 15, 16, 17, 32, 36]. In a policy-based architecture access requests

are evaluated against a predefined set of access control policies; while in hybrid architectures authorization tokens are issued based on the evaluation of access control policies. The only frameworks that do not fully satisfy the requirement is the one by Fremantle et al. [10] and Cirani and Picone [5]. The framework proposed in [10] is based on the OAuth protocol, which requires the resource owner to grant access to the application the first time an authorization token is issued. Similarly, Cirani and Picone [5] require the resource owner's involvement in the issuing of tokens. In particular, they account for three operational modes to obtain the tokens: *owner-to-owner*, in which a user registers his/her own device and obtains a token with all permissions on the device; *reactive owner-to-any*, in which the owner grants permission upon a user's request; and *proactive owner-to-any*, in which the owner proactively grants permission to a user.

Another key requirement for authorization frameworks designed for IoT applications is that they should not introduce communication and computation overhead on resource-constrained devices.

This requirement is typically addressed by outsourcing the most computationally expensive operation, namely policy evaluation, to an external service while performing only the enforcement of the access decision on constrained devices. Our analysis shows that most of the frameworks that adopt a policy-based architecture [1, 2, 3, 4, 7, 8, 9, 20, 22, 26, 28, 33, 40] externalize the PDP and the PAP (i.e., these components are not deployed in the physical node), thus fully satisfying the requirement. The only exceptions are the frameworks proposed in [12, 41] in which the PDP and the PAP run on the constrained device. Similarly, frameworks that rely upon a token-based or a hybrid architecture [6, 14, 15, 16, 17, 32, 36] fully satisfy the requirement. These frameworks employ dedicated services for the generation and issue of authorization tokens, and only the validation of the token is performed on the device. However, an aspect that should be considered is the size and format of the authorization token that could introduce a computation overhead on a constrained device. Lightweight standards to represent tokens like JSON should be preferred over XML-based formats like the one supported by SAML.

On the other hand, the performance of an access control system not only depends on the location of the components involved in the policy evaluation and communication protocol, but also on the policy evaluation strategy.

Frameworks that use an off-line evaluation strategy (i.e., [10, 32]) or an hybrid strategy in which only context constraints are verified at run-time (hybrid^c) and their verification does not require retrieving information from other components (e.g., [12, 15, 36, 41]), do not introduce latency in the access decision making process. Similarly, latency is limited if policy evaluation is performed on the edge like in [4, 16, 19, 40]. On the other hand, policy-based frameworks in which the access control mechanism is deployed in the cloud or provided as an external service (e.g., [1, 2, 3, 9, 18, 26, 42]) might introduce delay due to additional communication. This is also the case of frameworks that require validating tokens at run-time (hybrid^t) like in [5, 6, 14], or that require retrieving contextual information from external sources or from the cloud like in [18, 21, 33, 34, 42]. On top of this, the communication protocol has a significant impact on the overall performance of the IoT ecosystem where frameworks based on lightweight protocols like MQTT and CoAP provide better performance compared to the ones based on HTTP. Frameworks based on blockchain technology (e.g., [7, 8, 28, 29]) also do not satisfy the requirement due to time required to confirm a transaction. Every time an access control policy has to be added to or retrieved from the blockchain, a new transaction has to be created and added to the blockchain. Before a transaction can be added to the blockchain, special nodes

called miners run a consensus protocol that requires them to verify each transaction. The time to complete the validation process is typically in the order of minutes [27], which is clearly unsuitable for most IoT applications, especially for the ones that are latency sensitive.

Other key requirements for policy evaluation are interoperability and reliability/availability of components involved in the evaluation of the policies. However, despite their importance these two requirements are only marginally considered by existing authorization frameworks for IoT. Some of the frameworks only scratch the surface of the interoperability problem because they use a standard like XACML to specify the access control policies [36] or they encode the capability token in JSON [10, 15, 16, 18, 36]. Interestingly, Seitz et al. [36] provide an encoding of SAML assertions in JSON, while the others propose an ad-hoc format to encode tokens. However, using a standard only facilitates the exchange of policies or tokens across multiple domains but not their interpretation. If different authorities define their policies based on different semantic models, the collaborative evaluation of these policies can result in granting access to users for which access should be denied.

Reliability and availability is *fully satisfied* by those frameworks that can tolerate the failure of an architectural component and of the communication among them. Most of the frameworks partially satisfy the requirement because they only address the reliability/availability of the components but not of the communication among them. To address the failure of an architectural component, three main solutions have been adopted by existing authorization frameworks. Some frameworks have deployed the components in the cloud [1, 2, 17, 26, 33, 36], which guarantees that the components are evenly distributed across different servers, which are connected to work as one. Therefore, if one server fails, downtime is avoided. Salonikias et al. [33] instead ensure reliability and availability by replicating the PDP and the PEP and by defining propagation policies that specify how access control policies should be exchanged between PDPs. Frameworks based on blockchain [7, 8, 28, 29] propose to deploy and maintain a copy of the components of the authorization framework in all nodes forming the blockchain, thus ensuring resilience against failures of architecture components. The only framework proposed that fully satisfies is the one proposed by Neisse et al. [26], which adopts a reliable communication protocol like MQTT besides addressing the reliability of the architectural components.

An important aspect is the *applicability* of an access control framework to real IoT applications. In this respect, most of the proposed frameworks [1, 2, 3, 7, 8, 9, 13, 14, 17, 20, 28, 32, 33, 35, 41] only present the architecture of the access control mechanism and demonstrate the authorization flow among the components based on a realistic IoT use cases. For example, Dorri et al. [32] have illustrated their access control framework based on a smart home scenario. However, use cases do not provide insights on the effectiveness of the framework in realistic IoT settings. Only implementing the framework on a real IoT system and evaluating its performance and usability can provide such insights. Nonetheless, only few of the proposed frameworks have been implemented and evaluated [6, 12, 22, 23, 26, 36], while other works only report a prototype implementation [4, 10, 15, 16, 18, 19]. For instance, Neisse et al. [26] have proposed an authorization framework for MQTT brokers. The enforcement of access control policies is performed by a PEP that is integrated into the browser, while policy evaluation is done by an external PDP and Context Manager. The MQTT broker has been implemented using the Mosquitto library and its performance evaluated in terms of overhead introduced in the communication by implementing the PEP in the MQTT broker. Cirani et al.
citecirani2015iot have instead focused on evaluating the performance of their access control framework on constrained devices. In particular, they evaluated the energy and memory consumption of policy evaluation on a Contiki-based devices. To run the evaluation, they used the

Cooja simulator and considered Zolertia Z1 nodes with 92KB ROM and 8kb RAM. Similarly, Garcia et al. [12] have evaluated the performance of their framework on constrained devices but using a real testbed rather than a simulation environment like Cooja. The testbed consisted of Arduino Mega 2560 board3 with 16 MHz processor, 256 kB of Flash Memory, 8 kB of SRAM, and 4 kB of EEPROM.

5 Conclusion

In this deliverable we elaborate the software components of different cloud middleware within the network technologies used to continue the development of the use-cases selected within WP2 described in D2.1 and started in D2.2, we performed an extensive study using multiple testbeds and simulations to understand the influence of cloud technologies on network environment, in addition we evaluated cloud vs. edge behavior by enabling different V2C/V2E protocol relying on multiple cloud services(Authorization framework, SoTa provider, etc.,).

Bibliography

[1] Alshehri, Asma ; Sandhu, Ravi: Access Control Models for Cloud-Enabled Internet of Things: A Proposed Architecture and Research Agenda. In: *Proceedings of International Conference on Collaboration and Internet Computing*, IEEE, 2016, P. 530–538

[2] Alshehri, Asma ; Sandhu, Ravi: Access Control Models for Virtual Object Communication in Cloud-Enabled IoT. In: *Proceedings of International Conference on Information Reuse and Integration*, IEEE, 2017, P. 16–25

[3] Barka, Ezedine ; Mathew, Sujith S. ; Atif, Yacine: Securing the Web of Things with Role-Based Access Control. In: *Codes, Cryptology, and Information Security*, Springer, 2015, P. 14–26

[4] Bouij-Pasquier, Imane ; El Kalam, Anas A. ; Ouahman, Abdellah A. ; De Montfort, Mina: A Security Framework for Internet of Things. In: *Cryptology and Network Security*, Springer, 2015, P. 19–31

[5] Cirani, S. ; Picone, M.: Effective authorization for the Web of Things. In: *Proceedings of World Forum on Internet of Things*, IEEE, 2015, P. 316–320

[6] Cirani, Simone ; Picone, Marco ; Gonizzi, Pietro ; Veltri, Luca ; Ferrari, Gianluigi: IoT-OAS: An OAuth-Based Authorization Service Architecture for Secure Services in IoT Scenarios. In: *IEEE Sensors Journal* 15 (2015), Nb. 2, P. 1224–1234

[7] Dorri, Ali ; Kanhere, Salil S. ; Jurdak, Raja: Blockchain in Internet of Things: Challenges and Solutions / arXiv.org. 2016 (1608.05187). – arXiv:

[8] Dorri, Ali ; Steger, Marco ; Kanhere, Salil S. ; Jurdak, Raja: Blockchain: A distributed solution to automotive security and privacy. In: *IEEE Communications Magazine* 55 (2017), Nb. 12, P. 119–125

[9] Fernandez, Federico ; Alonso, Alvaro ; Marco, Lourdes ; Salvachua, Joaquin: A model to enable application-scoped access control as a service for IoT using OAuth 2.0. In: *Proceedings of Conference on Innovations in Clouds, Internet and Networks*, IEEE, 2017, P. 322–324

[10] Fremantle, Paul ; Aziz, Benjamin ; Kopecky, Jacek ; Scott, Philip: Federated Identity and Access Management for the Internet of Things. In: *Proceedings of International Workshop on Secure Internet of Things*, IEEE, 2014, P. 10–17

[11] Garcia Lopez, Pedro ; Montresor, Alberto ; Epema, Dick ; Datta, Anwitaman ; Higashino, Teruo ; Iamnitchi, Adriana ; Barcellos, Marinho ; Felber, Pascal ; Riviere, Etienne: Edge-centric Computing: Vision and Challenges. In: *SIGCOMM Comput. Commun. Rev.* 45 (2015), Nb. 5, P. 37–42

[12] Garcia-Morchon, Oscar ; Wehrle, Klaus: Modular context-aware access control for medical sensor networks. In: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2010, P. 129–138

[13] Guoping, Zhang ; Wentao, Gong: The Research of Access Control Based on UCON in the Internet of Things. In: *Journal of Software* 6 (2011), Nb. 4, P. 724–731

[14] Gusmeroli, Sergio ; Piccione, Salvatore ; Rotondi, Domenico: A capability-based security approach to manage access control in the Internet of Things. In: *Mathematical and Computer Modelling* 58 (2013), Nb. 5, P. 1189–1205

[15] Hernandez-Ramos, Jose L. ; Jara, Antonio J. ; Marin, Leandro ; Skarmeta, Antonio F.: Distributed capability-based access control for the Internet of Things. In: *Journal of Internet Services and Information Security* 3 (2013), Nb. 3/4, P. 1–16

[16] Hussein, Dina ; Bertin, Emmanuel ; Frey, Vincent: A Community-Driven Access Control Approach in Distributed IoT Environments. In: *IEEE Communications Magazine* 55 (2017), Nb. 3, P. 146–153

[17] Islam, S. M. R. ; Hossain, M. ; Hasan, R. ; Duong, T. Q.: A conceptual framework for an IoT-based health assistant and its authorization model. In: *Proceedings of Annual Computing and Communication Workshop and Conference*, IEEE, 2018, P. 616–621

[18] Jindou, J. ; Xiaofeng, Q. ; Cheng, C.: Access Control Method for Web of Things Based on Role and SNS. In: *Proceedings of International Conference on Computer and Information Technology*, IEEE, 2012, P. 316–321

[19] Kim, J. E. ; Boulos, G. ; Yackovich, J. ; Barth, T. ; Beckel, C. ; Mosse, D.: Seamless Integration of Heterogeneous Devices and Access Control in Smart Homes. In: *Proceedings of International Conference on Intelligent Environments*, IEEE, 2012, P. 206–213

[20] Kim, T. H. J. ; Bauer, L. ; Newsome, J. ; Perrig, A. ; Walker, J.: Access right assignment mechanisms for secure home networks. In: *Journal of Communications and Networks* 13 (2011), Nb. 2, P. 175–186

[21] Kim, Tiffany Hyun-Jin ; Bauer, Lujo ; Newsome, James ; Perrig, Adrian ; Walker, Jesse: Challenges in Access Right Assignment for Secure Home Networks. In: *Proceedings of USENIX Conference on Hot Topics in Security*, USENIX Association, 2010, P. 1–6

[22] Lee, Sanghak ; Choi, Jiwon ; Kim, Jihun ; Cho, Beumjin ; Lee, Sangho ; Kim, Hanjun ; Kim, Jong: FACT: Functionality-centric Access Control System for IoT Programming Frameworks. In: *Proceedings of Symposium on Access Control Models and Technologies*, ACM, 2017, P. 43–54

[23] Mahalle, P. N. ; Thakre, P. A. ; Prasad, N. R. ; Prasad, R.: A fuzzy approach to trust based access control in internet of things. In: *Proceedings of International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems*, IEEE, 2013, P. 1–5

[24] Mahalle, Parikshit N. ; Anggorojati, Bayu ; Prasad, Neeli R. ; Prasad, Ramjee: Identity Authentication and Capability Based Access Control (IACAC) for the Internet of Things. In: *Journal of Cyber Security and Mobility* 1 (2013), Nb. 4, P. 309–348

[25] Mell, Peter ; Grance, Tim: The NIST Definition of Cloud Computing / NIST. 2011 (800-145). – SP

[26] Neisse, Ricardo ; Steri, Gary ; Baldini, Gianmarco: Enforcement of security policy rules for the internet of things. In: *Proceedings of International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, 2014, P. 165–172

[27] Ouaddah, Aafaf ; Elkalam, Anas A. ; Ouahman, Abdellah A.: FairAccess: a new Blockchain-based access control framework for the Internet of Things. In: *Security and Communication Networks* 9 (2016), Nb. 18, P. 5943–5964

[28] Ouaddah, Aafaf ; Elkalam, Anas A. ; Ouahman, Abdellah A.: Towards a novel privacy-preserving access control model based on blockchain technology in IoT. In: *Europe and MENA Cooperation Advances in Information and Communication Technologies*. Springer, 2017 (Advances in Intelligent Systems and Computing 520), P. 523–533

[29] Pinno, O. J. A. ; Gregio, A. R. A. ; Bona, L. C. E. D.: ControlChain: Blockchain as a Central Enabler for Access Control Authorizations in the IoT. In: *Proceedings of Global Communications Conference*, IEEE, 2017, P. 1–6

[30] Ray, I. ; Alangot, B. ; Nair, S. ; Achuthan, K.: Using Attribute-Based Access Control for Remote Healthcare Monitoring. In: *Proceedings of International Conference on Software Defined Systems*, IEEE, 2017, P. 137–142

[31] Razzaque, Mohammad A. ; Milojevic-Jevric, Marija ; Palade, Andrei ; Clarke, Siobhán: Middleware for Internet of Things: a survey. In: *IEEE Internet of Things Journal* 3 (2016), Nb. 1, P. 70–95

[32] Rivera, Diego ; Cruz-Piris, Luis ; Lopez-Civera, German ; Hoz, Enrique de la ; Marsa-Maestre, Ivan: Applying an Unified Access Control for IoT-based Intelligent Agent Systems. In: *Service-Oriented Computing and Applications (SOCA), 2015 IEEE 8th International Conference on*, IEEE, 2015, P. 247–251

[33] Salonikias, Stavros ; Mavridis, Ioannis ; Gritzalis, Dimitris: Access control issues in utilizing fog computing for transport infrastructure. In: *CRITIS*, Springer, 2015 (LNCS 9578), P. 15–26

[34] Schuster, Roei ; Shmatikov, Vitaly ; Tromer, Eran: Situational Access Control in the Internet of Things. In: *Proc. of CCS*, ACM, 2018, P. 1056–1073

[35] Sciancalepore, S. ; Piro, G. ; Tedeschi, P. ; Boggia, G. ; Bianchi, G.: Multi-Domain Access Rights Composition in Federated IoT Platforms. In: *Proceedings of Workshop on Recent Advances in Secure Management of Data and Resources in the IoT*, 2018

[36] Seitz, Ludwig ; Selander, Goran ; Gehrmann, Christian: Authorization Framework for the Internet-of-Things. In: *Proceedings of International Symposium on A World of Wireless, Mobile and Multimedia Networks*, IEEE, 2013, P. 1–6

[37] Sethi, Pallavi ; Sarangi, Smruti R.: Internet of Things: Architectures, Protocols, and Applications. In: *Journal of Electrical and Computer Engineering* 2017 (2017)

[38] Song, Zhexuan ; Cardenas, Alvaro A. ; Masuoka, Ryusuke: Semantic middleware for the Internet of Things. In: *Proceedings of International Conference on the Internet of Things*, IEEE, 2010, P. 1–8

[39] Stojmenovic, Ivan ; Wen, Sheng: The fog computing paradigm: Scenarios and security issues. In: *Proceedings of Federated Conference on Computer Science and Information Systems*, IEEE, 2014, P. 1–8

[40] Tian, Yuan ; Zhang, Nan ; Lin, Yueh-Hsun ; Wang, XiaoFeng ; Ur, Blase ; Guo, Xianzheng ; Tague, Patrick: SmartAuth: User-Centered Authorization for the Internet of Things. In: *Proceedings of USENIX Security Symposium*, 2017, P. 361–378

[41] Ye, Ning ; Zhu, Yan ; Wang, Ru-chuan ; Malekian, Reza ; Qiao-min, Lin: An Efficient Authentication and Access Control Scheme for Perception Layer of Internet of Things. In: *Appl. Math* 8 (2014), Nb. 4, P. 1617–1624

[42] Zhang, Guoping ; Tian, Jiazheng: An extended role based access control model for the Internet of Things. In: *Proceedings of International Conference on Information, Networking and Automation* Volume 1, IEEE, 2010, P. 319–323