H2020 5G-Coral Project

Grant No. 761586

D3.1 – Initial design of 5G-CORAL orchestration and control system

Abstract

This deliverable presents the initial design of the 5G-CORAL orchestration and control system, namely OCS. Opportunities and requirements are identified for a joint edge and fog orchestration system. The OCS architecture and its design approach, along with selected components and their interfaces, are described next with emphasis on mobility and volatility of resources. Approaches for resource discovery and integration are described according to the defined 5G-CORAL use cases.

Document properties

Document number	D3.1
Document title	Initial design of 5G-CORAL orchestration and control system
Document responsible	TELCA
Document editor	José María Roldán Gil
Target dissemination level	Public
Status of the document	Stable
Version	1.0

Table of contributors

Partner	Contributors
UC3M	Luca Cominardi, Iñaki Úcar Marqués
IDCC	Giovanni Rigazzi, Charles Turyagyenda
TELCA	José María Roldán Gil, Aitor Zabala Oribe
ITRI	Robert Gdowski, Samer Talat, Shahzoob Bilal Chundrigar, Kuei-Li Huang, Ibrahiem Osamah
ADLINK	Gabriele Baldoni
NCTU	Li-Hsing Yen, Hojjat Baghban

Production properties

Reviewers	Carlos J. Bernardos, Alain Mourad, Shahzoob Bilal Chundrigar

Document history

Revision	Date	Issued by	Description
1.0	27 May 2018	TELCA	D3.1 ready for publication

Disclaimer

This document has been produced in the context of the 5G-Coral Project. The research leading to these results has received funding from the European Community's H2020 Programme under grant agreement N° H2020-761586.

Table of Contents

List of Figures

List of Tables

List of Acronyms

3DSP	3D Synchronisation Profile
3GPP	3rd Generation Partnership Project
5G-NR	5G New Radio
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
A2DP	Advanced Audio Distribution Profile
ACK	Acknowledgement
ACRN	
AMQP	Advanced Message Queuing Protocol
ANDI	Access Network Discovery Information
ANDSF	Access Network Discovery and Selection Function
ANQP	Access Network Query Protocol
AP	Access Point
API	Application Programming Interface
APP	Application
AR	Augmented Reality
ARP	Address Resolution Protocol
ATM	Asynchronous Transfer Mode
AVRCP	Audio/Video Remote Control Profile
BBU	Base-Band Unit
BDDP	Broadcast Domain Discovery Protocol
BIOS	Basic Input/Output System
BLE	Bluetooth Low Energy
BPP	Basic Printing Profile
BSS	Business Support System
CADS	Coordinated Application Deployment System
CAN-BUS	Controller Area Network BUS
CD	Computing Devices
CDP	Cisco Discovery Protocol
COAP	Constrained Application Protocol
CP	Control Plane
CPU	Central Processing Unit
C-VLAN	Customer Virtual Local Address Network
D2D	Device to Device
DA	Dictionary Agents
DC	Data Centre
DDS	Data Distribution Service
DHCP	Dynamic Host Configuration Protocol
DIMM	Dual In-line Memory Module
DNS	Domain Name System
DPI	Deep Packet Inspection
DPID	Data Path ID
DRAM	Dynamic Random Access Memory
DSRC	Dedicated Short-Range Communications
EAP	Extensible Authentication Protocol
EAS	Emergency Alert System
EFS	Edge and Fog computing System
eNB	Evolved Node B
EPC	Evolved Packet Core
ePDG	Evolved Packet Data Gateway
ESS	Extended Service Set
ETSI	European Telecommunications Standards Institute
exFAT	EXtended File Allocation Table
FAT32	File Allocation Table 32
FMC	Follow-Me Cloud
FSM	Finite State Machine
FTP	File Transfer Protocol
GAS	Generic Advertisement Service
GENA	General Event Notification Architecture
gNB	Next Generation NodeB
GPIO	General Purpose Input/Output
GPRS	General Packet Radio Service
GPS	Global Positioning System
GPU	Graphics Processing Unit
GTP	GPRS Tunnelling Protocol
GUI	Graphical User Interface
H-ANDSF	Home Access Network Discovery and Selection Function
HDD	Hard Disk Drive
HDP	Health Device Profile
HESSID	Homogeneous ESS IDentifier
HFP	Hands-Free Profile
HPLMN	Home-PLMN
HSP	HeadSet Profile
HST	High Speed Train
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
HV	Hypervisor
HW	HardWare
I/O	Input/Output
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISG	Industry Specification
IS-IS	Intermediate System to Intermediate System
ISMP	Inter System Mobility Policies
ISRP	Inter System Routing
JDBC	Java DataBase Connectivity
JMX	Java Management eXtensions
JSON	JavaScript Object Notation
KPI	Key Performance Indicator
KVM	Kernel-based Virtual Machine
L2CAP	Logical Link Control and Adaptation Protocol
LAN	Local Address Network
LLC	Logical Link Control
LLDP	Link Layer Discovery Protocol
LLDPDU	Link Layer Discovery Protocol Data Unit
LLMNR	Link-Local Multicast Name Resolution
LTE	Long Term Evolution
LXC	LinuX Containers
LXD	Next generation system container manager
MAC	Media Access Control
MANO	MANagement and Orchestration
MAP	Message Access Profile
MEC	Mobile Edge Computing
MED	Media Endpoint Discovery
MIB	Management Information dataBase
MIH	Media Independent Handover
MIP	Mobile IP
MO	Manageable Object
MPLS	MultiProtocol Label Switching
MQTT	Message Queuing Telemetry Transport
MSAP	MAC Service Access Point
MTU	Maximum Transmission Unit
N3IWF	Non-3GPP InterWorking Function
NAI	Network Access Identifier
NAS	Non-Access Stratum

NAT	Network Address Translation
NB-IoT	NarrowBand IoT
NDP	Neighbour Discovery Protocol
NETCONF	NETwork CONFiguration protocol
NFS	Network File System
NFV	Network Function Virtualisation
NG	Next Generation
NRPE	Nagios Remote Plugin Executor
NS	Network Service
NSD	Network Service Descriptor
OBU	On Board Unit
OCS	Orchestration and Control System
OI	Organisation Identifier
OMA-DM	Open Mobile Alliance Device Management
ONOS	Open Network Operating System
OPP	Object Push Profile
OS	Operating System
OSPF	Open Shortest Path First
OSS	Operation Support System
OVS	Open Virtual Switch
OVSDB	Open vSwitch Database Management Protocol
P2P	Peer to Peer
PAN	Personal Area Networking Profile
PBAP	Phonebook Access Profile
PCI	Peripheral Component Interconnect
PCP	Performance Co-Pilot
PDN	Packet Data Network
P-GW	Packet Data Network Gateway
PHY	PHYsical layer
PLMN	Public Land Mobile Network
PMDA	Performance Metrics Domain Agent
PMIP	Proxy Mobile IP
PNF	Physical Network Functions
PoE	Power over Ethernet
POP3	Post Office Protocol
PpU	Pay per Use
PTR	PoinTer Record
PXE	Preboot eXecution Environment
QoS	Quality of Service
QPI	Quick Path Interconnect
RADIUS	Remote Authentication Dial-In User Service
RAID	Redundant Array of Independent Disks
RAM	Random Access Memory
RAN	Radio Access Network
RAT	Radio Access Technologies
RLQP	Registered Location Query Protocol
ROM	Read-Only Memory
RPC	Remote Procedure Call
RSU	Road Side Unit
RU	Request Unit
SA	Service Agents
SAN	Storage Area Network
SAP	SIM Access Profile
SD	Secure Digital
SDN	Software Defined Network
SDP	Service Discovery Protocol
SIG	Special Interest Group
SLA	Service Level Agreement
SLP	Service Location Protocol
SMS	Short Message Service
SMTP	Simple Mail Transfer Protocol
SNAP	Subnetwork Access Protocol
SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
SoC	System on Chip
SPP	Serial Port Profile
SRAM	Static Random Access Memory
SR-IOV	Single-Root Input/Output Virtualisation
SRV	Service Resource Records
SSD	Static Solid Disk
SSDP	Simple Service Discovery Protocol
SSID	Service Set IDentifier
SSLP	Simple Service Location Protocol
SSPN	Subscription Service Provider Network
S-VLAN	Service VLAN
TA	Translation Agents
TCP	Transmission Control Protocol
TDBaaS	Time-series Database as a Service
TDLS	Tunnelled Direct Link Setup
TDP	Thermal Design Power
TFTP	Trivial file transfer Protocol
TLS	Transport Layer Security
TLV	Type-Length-Value
TOSCA	Topology and Orchestration Specification for Cloud Applications
TPMR	Two Port MAC Relay
TWAG	Trusted Wireless Access Gateway
TXT	Text Record
UA	User Agent
UDP	User Datagram Protocol
UE	User Equipment
UNDI	Universal Network Device Interface
UP	User Plane
UPnP	Universal Plug and Play
URI	Uniform Resource Identifiers
URL	Uniform Resource Locator
USN	Unique Service Name
UUID	Universally Unique IDentifier
V2I	Vehicle to Infrastructure
V2X	Vehicle to Anything
V-ANDSF	Visiting Access Network Discovery and Selection Function
vCPU	Virtual Central Processing Unit
VDP	Video Distribution Profile
VDU	Virtualisation Deployment Unit
VIM	Virtualisation Infrastructure Managers
VLAN	Virtual Local Address Network
VLD	Virtual Link
VM	Virtual Machine
VNF	Virtual Network Functions
VNFC	VNF Components
VNFD	VNF Descriptors
VNFM	VNF Manager
VoIP	Voice over IP
VPLMN	Visited-PLMN
VR	Virtual Reality
VTP	VLAN Trunking Protocol
WAN	Wide Area Network
WAP	Wireless Application Protocol
WBXML	WAP Binary XML
WLAN	Wireless Local Area Network
WMI	Windows Management Instrumentation
WP	Work Package
XML	eXtensible Markup Language
XMP	Extensible Metadata Platform
ZDP	ZigBee Device Profile

Executive Summary

This first deliverable from 5G-CORAL Work Package 3 focuses on the design of the Orchestration and Control System (OCS). It first identifies the opportunities for joint edge and fog orchestration which serve as the basis for defining next the OCS requirements and architecture. Emphasis is put on the dynamicity of 5G-CORAL resources which can be mobile and volatile and can communicate via multiple access technologies. Based on the architecture and formulated requirements, the OCS design choices are elaborated aiming at the resources discovery and integration, including federation aspects.

The key achievements in this deliverable are highlighted below:

- Identification of OCS opportunities and potential benefits. These have been recently published in [55].
- Identification of OCS functional and non-functional requirements [56], along with a deep-dive into the OCS architecture listing all the different interfaces identified;
- Elaboration of OCS design choices based on the requirements and architecture, where the following topics are explored: heterogeneous resources support, dynamic resources support, dynamic migration support, monitoring support, and third-parties support;
- Proposal of a baseline solution for resource discovery and integration across multiple access technologies, such as IEEE 802.11, 3GPP, Bluetooth/ZigBee, and Ethernet, and IPv6 networks [57][58];
- Characterisation of the integration of the OCS and the Edge and Fog System (EFS) where the options of trusted/untrusted resources and pricing are accounted for;
- First OCS prototype publicly released as open source [59].

1 Introduction

To increase flexibility in service offerings and network management, the European Telecommunications Standards Institute (ETSI) Network Function Virtualisation (NFV) Industry Specification Group (ISG) pioneered the idea of bringing virtualisation capabilities into mobile operator networks [4]. By decoupling the network functions from the underlying hardware platform, NFV allows operators to dynamically deploy services in response to the needs of the traffic and customers. In addition to NFV, ETSI Multi-access Edge Computing (MEC) ISG brings computing capabilities close to the end users to cope with the ever-increasing amount of data (e.g., generated by IoT) and the low latency required by some use cases (e.g., vehicular communication) [5]. NFV and MEC jointly represent a paradigm shift for mobile operator networks, which evolve from a centralised architecture based on monolithic and hardware-integrated functions to a software-based distributed architecture. Such evolution enables a common hosting environment, namely edge computing, at the network edge characterised by low latency and high bandwidth as well as real-time access to radio network information. Network functions and software applications can hence be deployed close to the end users, thus alleviating congestion at the mobile network core and serving efficiently local purposes, such as data aggregation for IoT, localised real-time control, and single aggregation point for multi-access connectivity.

Recently, fog computing gained considerable traction in the industrial community as demonstrated by the launch of OpenFog consortium [6]. Fog computing distributes computing, storage, control and networking functions closer to the users along a cloud-to-thing continuum which also envisions the collaborative usage of a multitude of the end user or near-user edge devices to carry out a substantial amount of those tasks. It is noteworthy that non-stationary and volatile devices are also considered in fog computing, for example when apparatuses are hosted on moving devices (e.g., car, train, mobile user) or are battery-powered (e.g., IoT). While edge computing focuses on operator networks and related use cases, fog computing focuses more broadly on enterprise use cases, which may not be necessarily related to mobile networks (e.g., smart cities, remote surveillance, etc.). Nonetheless, edge and fog share a common goal: that is to bring networking and computing capabilities closer to the end user. Nowadays, edge and fog computing are stand-alone domains that require separate deployments eventually contending for the same physical resources (e.g., spectrum). The lack of integration poses numerous challenges to the effective usage of those resources in addition to the cost-effectiveness of having multiple separate physical deployments [7].

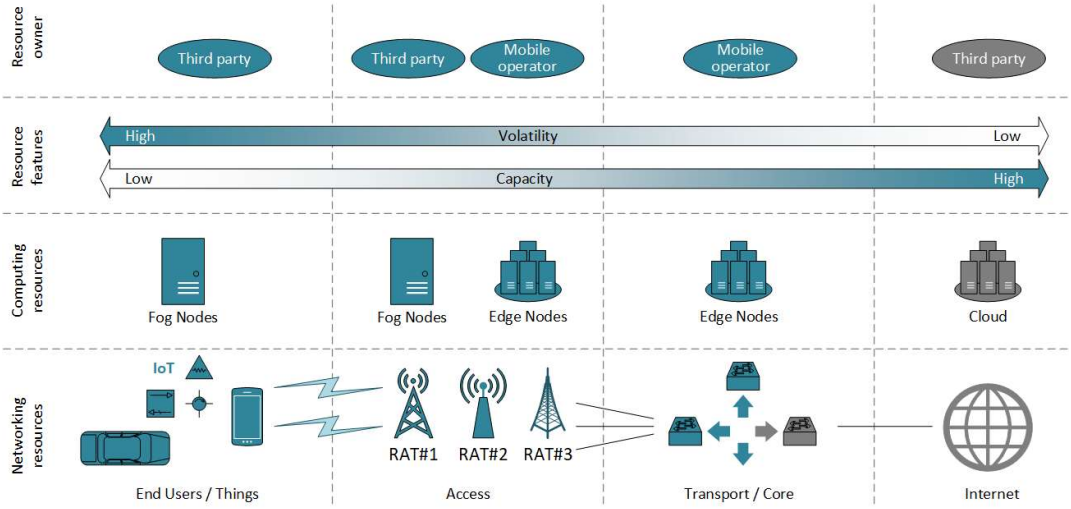

FIGURE 1-1: EDGE AND FOG RESOURCES AND CHARACTERISTICS

Integrating resources belonging to distinct administrative domains is a challenge that goes beyond the pure technological dimension and involves trust relationships between parties. To this end,

federation provides the means for integrating multiple administrative domains at different granularity into a unified platform where the federated resources can trust each other at a certain degree, whereas the federation trust is the embodiment of a service/business-level agreement or partnership between two organisations [8]. Figure 1-1 shows the edge and fog resources (in blue) which may be federated among themselves and interact with centralised core and cloud domains (in grey) for offering a real cloud-to-thing continuum.

This deliverable presents the initial design of the 5G-Coral OCS. It is structured as follows:

Section 2 first presents the opportunities together with the functional and non-functional requirements of the targeted OCS. Secondly, it presents an ETSI NFV-compliant architecture for the OCS and identifies the gaps in existing approaches and components.

Section 3 elaborates on the design choices for the OCS components and subsystems. Support of heterogeneous and dynamic resources is described as well as the monitoring and integration with third-parties considering federation and pricing aspects.

Section 4 provides a deep dive into the procedures for discovering and integrating Edge and Fog resources into a target EFS. Multi-Radio Access Technology (RAT) aspects are addressed as to allow the discovery and integration of the resources across multiple access technologies.

Section 5 draws a conclusion to summarise the findings of this deliverable and outlines the next steps and the prospects for future work.

Appendix A, **Appendix B** present a detailed description of the state-of-art on discovery protocols and monitoring tools, respectively. Finally, **Appendix C** presents the trusted and untrusted network access in 3GPP networks.

2 OCS Benefits, Requirements, and Architecture

By bringing the fog into the vision of 5G and beyond, several opportunities can be anticipated to enhance the system efficiency and performance as presented in Section 2.1. The identified benefits are used next to define the OCS requirements in Section 2.2. Finally, Section 2.3 describes the OCS architecture and components.

2.1 OCS Benefits

The broad necessity of cloud service providers to fulfil their sparse customers, their needs and concerns in line with customer's data ownership and scarcity of standards defining inter-clouds service interfaces, has led to the adoption of decentralised cloud federations. A cloud federation can integrate a pool of diverse services from multiple service providers that self-govern each other by using well-defined interfaces and agreements between them [8]. Cloud federation is a key enabling technology for cooperative service deployment. Dynamically, it allows heterogeneous and independently administrated clouds to interact and share resources with each other. Federated clouds offer an integrated cloud service by federating infrastructures provided by different cloud service providers. The ability of cloud federation to share cloud resources among participating service providers improves resource utilisation and enhances elasticity and reliability of cloud service. Federated clouds also enable new business opportunities.

Virtualisation technologies and its orchestration, including the use of virtual machines and containers, play a significant role in the provisioning of elastic mobile services in federated clouds. In this case, the federation mechanisms should include functionalities such as deployment, runtime management and monitoring, termination, authentication, access control and live migration of services in remote clouds [9]. Many existing works in the literature develop frameworks and architectures to enable provisioning and management of services in federated clouds. Depending on the cooperation model of participants, cloud federation can be classified into three types. The first one is a horizontal federation, where participants cooperate on a peer-to-peer basis. This type of federation applies well to the case of federated mobile edge systems. The second type is a vertical federation, where participants are entities in a hierarchy, like hybrid cloud [13][14] which combines the services provided by a private cloud and a third-party public cloud. This type of federated edge-and-fog architecture refers to the federation between edge and fog systems, between central cloud and edge system, or between central cloud and fog system. Finally, the third type of federation is a hybrid of both horizontal and vertical federations.

Most existing federated clouds fall into the category of the vertical federation. For instance, Follow-Me Cloud (FMC) [10] proposes an architecture for federated cloud and distributed mobile network environment which allows the services delivery through an optimal service anchor and the possibility of following mobile users as they roam through federated cloud environments. FMC utilises Markov-Decision-Process to make cost-effective and performance optimised migration decisions. Furthermore, challenges which cloud providers may face when participating in a federated cloud environment include the heterogeneity of cloud management systems and models describing the services. To resolve this issue, [11] proposes a coordinated application deployment system (CADS) to enable the description of the desired service deployment in the form of a topology model. In this way, CADS provides interoperability in the deployment of services in federated clouds.

The NFV ISG defines a Management and Orchestration (MANO) framework [4] for deploying network services in an NFV environment. Nowadays, NFV MANO scope is limited to a single mobile operator network. To overcome such limitation, an NFV Work Item has been recently approved with the aim of enabling the management and orchestration across multiple operators [12]. Although logical inter-connection between different mobile operators is being defined, integration with third-party domains (e.g., fog or cloud) is still not considered. Like NFV, MEC framework [5] only considers a single network operator domain and does not consider integration with third-party domains like fog. Finally, although ETSI MEC and NFV enable mobility of applications and

services, it is only within the boundaries of the stationary edge resources of the mobile operator and volatile resources are not considered.

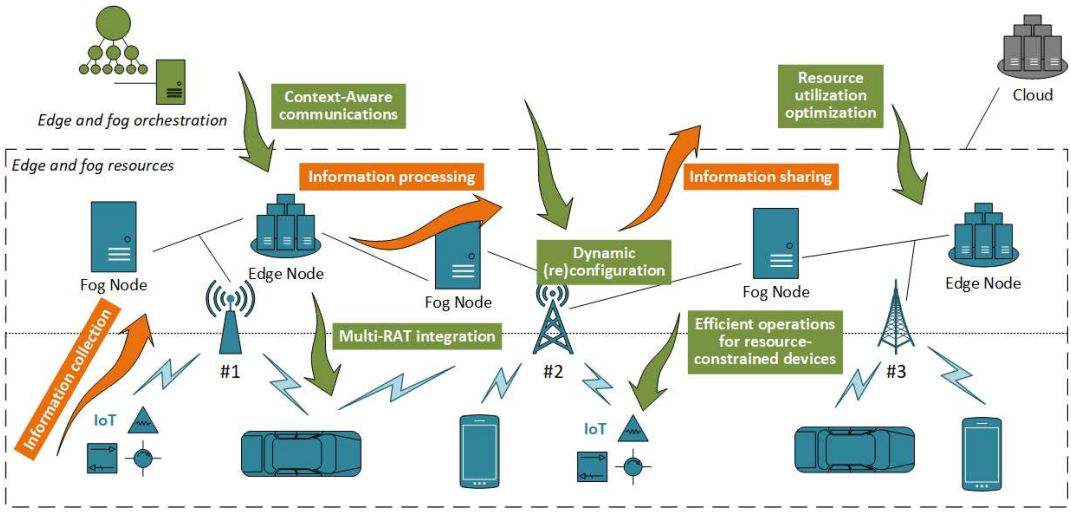

FIGURE 2-1: EDGE AND FOG JOINT ORCHESTRATION BENEFITS

Figure 2-1 illustrates the potential benefits of jointly orchestrating resources at the edge and the fog with the objective of overcoming the limitations presented above. Indeed, instead of solely relying on the computing substrates in the edge data centres, edge-and-fog orchestration creates a larger pool of resources distributed near the end users, including any type of devices that possess networking and computing capabilities. These range from vehicles and drones, to smartphones, tablets, and laptop computers, to IoT devices such as sensors or actuators. Clearly, some of these devices will have limited computational capability and battery (the so-called "resource-constrained" devices) due to their low-cost nature. Having such large pool of resources available near the user naturally enables low latency communication across multiple RATs by **distributing** the various computing and networking **tasks across both edge and fog resources**. Besides, the **resource-constrained devices** can now rely on the edge and fog resources to execute some of their computationally and power demanding tasks. This presents an opportunity for low-cost devices to remain intelligent despite their limited capabilities by leveraging advanced applications running on the EFS. This also leads to higher multiplexing gains and greater utilisation efficiency of the resources for executing certain functions or tasks tailored to the needs of the applications and end users. Such paradigm may create new business models wherein terminal devices can also participate in the pool of edge and fog resources in return of incentives (e.g., service subscription reduction), which helps infrastructure providers decreasing the deployment and maintenance cost of their edge data centres.

The great diversity of the resources presents a fantastic opportunity for collecting, processing, and sharing a multitude of information coming from the underlying infrastructure of computing, storage and networking resources (shown in orange in Figure 2-1). Such information potentially opens a new degree of freedom in (i) optimizing the network performance and (ii) lowering the network operational cost based on the extracted context information. **Context-aware communications** raise the opportunity of developing new algorithms to optimise the network performance based on the learning and intelligence derived from the context information of the edge-and-fog system. For example, where the edge-and-fog system is likely to have multiple co-existing RATs, one can envision efficient multi-RAT management and coordination algorithms by leveraging on the radio information extracted from each RAT. Moreover, context information can be used for **dynamic allocation of the computing and networking resources** to prioritise edge and fog resources in an area of higher demand which may lead to more optimised resource utilisation. A concentrated traffic or computing request can be directed to a limited number of edge and fog resources while others will shift to idle mode or be switched off, thus improving the overall energy conservation of the system. Furthermore, software migration and placement capability of the orchestration process

allows for a seamless transfer of intelligence between geographically disparate nodes. This tackles the variable application delay constraints. Software components can be placed in the user vicinity fulfilling its latency requirements. Together with the ability to create interrelation (chain) of functions and applications and then map it into an underlying substrate of computing and networking resources, such ecosystem will be able to handle any 3rd party driven dependency between functions and applications whilst preserving the scalability of the solution at the same time.

2.2 OCS Requirements

Based on the presented state of the art and the identified potential benefits of OCS (Section 2.1), we have performed an analysis leading to the definition of OCS functional and non-functional requirements in Table 2-1 and Table 2-2, respectively. These requirements are then mapped against the system-level requirements defined in [34] for the overall 5G-CORAL solution.

TABLE 2-1: OCS FUNCTIONAL REQUIREMENTS

ID	Requirement	System-level requirement
OF-01	Support of harvesting computing capabilities from low-end resources	FT-01, FT-02
OF-02	Support of harvesting computing capabilities from mobile resources	FT-01, FT-02, FT-09, FT-13
OF-03	Support of discovery, configuration, monitoring, allocation, etc. of relevant hardware capabilities (e.g., wireless interfaces, GPIO, GPU, SR-IOV, etc.)	FT-03, FT-06, FT-08, FT-13, FT-14, FT-15
OF-04	Support of integration including at runtime of heterogeneous resources in terms of software and hardware capabilities (e.g., different CPU arch, hypervisors, etc.)	FT-04, FT-05, FT-09, FT-10, FT-11, FT-12
OF-05	Support of federation including at runtime of OCS components	FT-05
OF-06	Support of the interworking with resources external to the OCS (e.g., cloud-to-thing continuum)	FT-06

TABLE 2-2: OCS NON-FUNCTIONAL REQUIREMENTS

ID	Requirement	System-level requirement
ON-01	Support of deployment of OCS on low end devices (e.g., battery-limited, form-factor, resource constrained, etc.)	NF-04
ON-02	Support of deployment of OCS on mobile devices (e.g., car, robot, train, etc.)	NF-04
ON-03	Availability and self-healing mechanisms in error-prone environments	NF-02, NF-09, NF-10
ON-04	Support of large deployments in terms of number of resources and geographic areas	NF-11
ON-05	Support of plugins for extensibility	NF-08
ON-06	Capability to adapt to workload changes by provisioning and de-provisioning resources in an automated manner	NF-04, NF-08
ON-07	Support of multiple tenants participating and co-existing in the same environment	NF-05, NF-06, NF-07, NF-12

2.3 OCS Architecture

For the sake of keeping this document self-contained, we first report the overall 5G-CORAL architecture which is specified in [7]. The 5G-CORAL architecture is based on the ETSI MEC and ETSI NFV frameworks, where a mix of physical and virtualised resources available in the fog and edge tiers is considered in order to implement an ETSI NFV compliant infrastructure, and contemplates two major building blocks:

- **Edge and Fog computing System (EFS):** an EFS is a logical system subsuming Edge and Fog resources that belong to a single administrative domain. An EFS provides service platforms, functions, and applications on top of available resources, and may interact with other EFS domains.
- **Orchestration and Control System (OCS):** an OCS is a logical system in charge of composing, controlling, managing, orchestrating, and federating one or more EFS(s). An OCS comprises Virtualisation Infrastructure Managers (VIMs), EFS managers, and EFS orchestrators. An OCS may interact with other OCS domains.

According to [33], an edge resource is a physical device that provides computing, storage and/or networking capabilities, hosted inside an edge data-centre and belongs to an EFS. A fog resource is a physical device that provides computing, storage and/or networking capabilities, considered to be end-user or near-user edge devices, belonging to an EFS. Usually its capabilities are more constrained than an edge resource. They might have a volatile nature expressed through low availability which influences design aspects such as dominance of wireless communication, battery support and low resource extensibility as in comparison Edge resources.

An EFS Service Platform is a logical data exchange platform within EFS consisted of (i) data storage to keep the collected information from applications/functions and edge/fog resources, and (ii) communication protocol to gather/provide information from/to applications/functions and edge/fog resources. A function is a software entity by at least one atomic entity deployed in EFS for network infrastructure. An application is a software entity comprised by at least one atomic entity deployed in EFS for users and third parties. Finally, an atomic entity is an unpartitionable computing task executed in the EFS. EFS Applications, Functions, and Service Platform are also referred to as EFS entities.

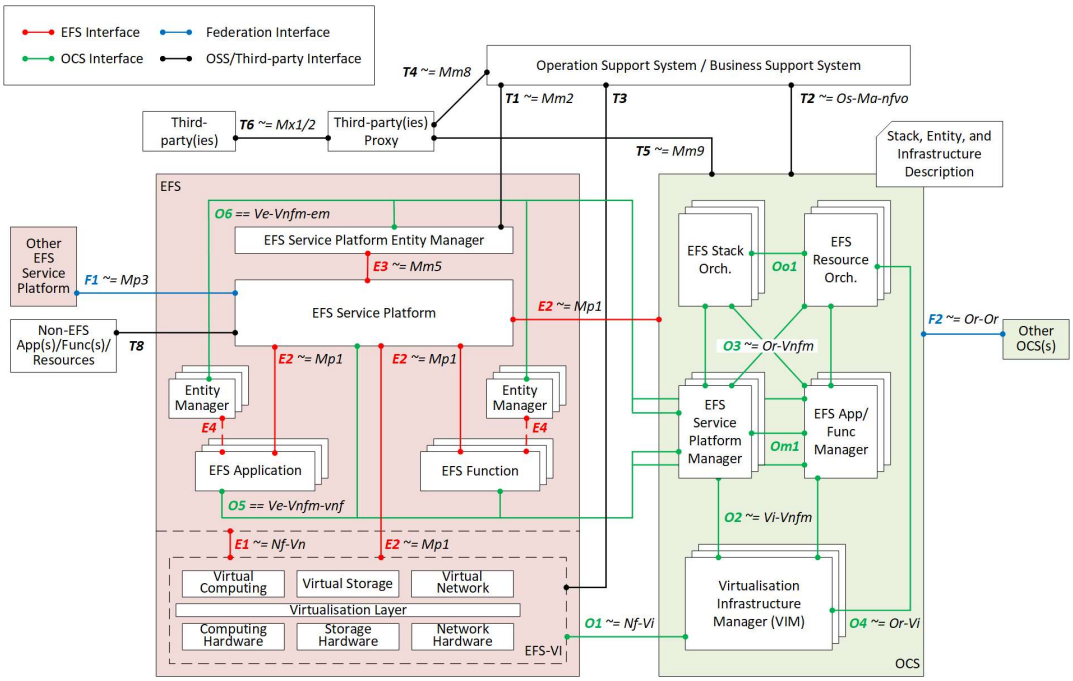

FIGURE 2-2: 5G-CORAL SYSTEM ARCHITECTURE

Figure 2-2 shows the refined 5G-CORAL architecture developed in collaboration with WP2. The main changes compared to the one presented by WP1 in [34] are the following:

- NFVI has been renamed as EFS-VI as to encompass the capability of the EFS Virtualisation Infrastructure to also host applications and service platform;

- The Element Managers in the EFS have been renamed as Entity Managers to reflect the definition of EFS Application, Function, and Service Platform. Nevertheless, the Entity Managers play the same role of ETSI NFV Element Managers as initially stated;
- T8 interface has been extended as to also support non-EFS Resources;
- O5 and O6 interfaces connect also the EFS App/Func Manager to the corresponding EFS Application/Function Entity Manager;
- VNF Descriptor has been changed to Entity Descriptor to cover EFS Application, EFS Function, and EFS Service Platform Descriptor.

In the following we detail the OCS components which are shown from bottom to top in Figure 2-2:

- A **Virtualisation Infrastructure Manager (VIM)** comprises the functionalities that are used to control and manage the interaction of the service platforms, functions, and applications with the edge and fog resources under its authority, as well as their virtualisation. Multiple VIMs may be deployed to control and manage distinct virtualisation substrates (e.g., virtual machine, containers) or administrative domains. The VIM exposes and makes use of the O1, O2, and O4 interfaces as well as the E2 interface;
- An **EFS Manager** is responsible for the lifecycle management (e.g. instantiation, update, query, scaling and termination) of the service platforms, functions, and applications in the EFS. Multiple EFS Managers may be deployed to manage distinct components of the EFS (e.g., service platforms, functions, and applications) whereas each EFS Manager may manage a single service platform, function, application or a pool of them. Given the inter-dependency of applications/functions and the service platform due to the publish-subscribe model [33], two EFS Managers are defined at architectural level: one dedicated to the management of the service platform and one dedicated to the management of the applications and functions. The EFS Managers expose and make use of the O2, O3, O5, O6 and, Om1 interfaces, as well as the E2 interface;
- An **EFS Orchestrator** is in charge of the orchestration and management of edge and fog resources and composing the EFS. An EFS Orchestrator comprises an EFS Resource Orchestrator and an EFS Stack Orchestrator. An EFS Resource Orchestrator supports accessing the edge and fog resources in an abstracted manner independently of any VIMs, as well as governance of service platform/function/application instances sharing resources in the EFS. An EFS Stack Orchestrator is responsible for the EFS Stack lifecycle management operations (e.g. instantiation, update, query, scaling and termination); The EFS Orchestrators expose and make use of the O3, O4, Oo1, T2, and T5 interfaces, as well as the E2 interface;
- An **EFS Stack** can be viewed architecturally as a forwarding graph of functions and/or application interconnected by supporting edge and fog resources and/or service platforms. An EFS Stack extends the ETSI NFV Network Services by also considering interconnections with applications and service platforms and not only between network functions;
- An **EFS Stack Descriptor** extends the ETSI NFV Network Service Descriptor by also considering applications and service platforms in addition to network functions. It describes the requirements and interconnections of one or more EFS Functions and EFS Applications between them or with the EFS Service Platform;
- An **EFS Entity Descriptor** extends and combines ETSI NFV VNF and ETSI MEC App descriptors as to uniformly describe the various characteristics of EFS Functions, EFS Applications, and EFS Service Platform. EFS Entity Descriptors are referenced and included into an EFS Stack Descriptor as to allow the OCS to properly deploy all the entities and interconnect them.

The role of each OCS interface and its relevance to ETSI NFV and MEC reference points are reported in Table 2-3.

TABLE 2-3: OCS INTERFACES

ID	ETSI NFV/MEC ref. point	Description
O1	ETSI NFV: Nf-Vi	This interface is used for exchanges between the VIM and the edge and fog resources composing the EFS and supports the following: • Allocation of functions and applications with indication of compute/storage resource; • Update, migration, and termination of functions and applications including their resource allocation; • Creation, configuration, and termination of connection between functions, applications, and service platform. Compared to ETSI NFV environment, such interface also needs to support intermittent connectivity on volatile low-end devices with heterogeneous virtualisation support.
O2	ETSI NFV: Vi-Vnfm	This interface is used for exchanges between the VIM and the EFS Managers and supports the following: • Edge and fog resources information retrieval; • Edge and fog resource allocation and release; • Notification from the VIM to the EFS Manager of events, measurement results, and usage records regarding edge and fog resources used by a specific application or function. Compared to ETSI NFV environment, such interface also needs to support information regarding mobility and battery-level of resources in addition to information regarding privacy constraints and negotiated SLAs.
O3	Or-Vnfm	This interface is used for exchange between EFS Orchestrator and EFS Manager, and supports the following: • Edge and fog resources authorisation, validation, reservation, and release for a function or application; • Edge and fog resources allocation/release request for an application or function; • Application and function instantiation, update, termination, scaling in/out and up/down; • Application and function instance query to retrieve any run-time information. Compared to ETSI NFV environment, such interface also needs to support information regarding federation and privacy constraints of underlying resources.
O4	ETSI NFV: Or-Vi	This interface is used for exchange between the VIM and the EFS Orchestrators and supports the following: • Edge and fog resources information retrieval; • Edge and fog resource allocation and release; • Function and application addition, deletion, update; • Notification from the VIM to the EFS Orchestrator of events, measurement results, and usage records regarding edge and fog resources.

ID	ETSI NFV/MEC ref. point	Description
		Compared to ETSI NFV environment, such interface also needs to support information regarding mobility and battery-level of resources in addition to information regarding privacy constraints and negotiated SLAs.
O5	ETSI NFV: Ve-Vnfm-Vnfm	This interface is used for exchange between functions or applications and the corresponding EFS App/Function Manager, or between the EFS Service Platform and the corresponding EFS Service Platform Manager, and supports the following: • Notification from the application/function/service platform to the corresponding EFS Manager of events, measurements, and usage records regarding the application/function/service platform itself; • Verification that the application or function is still alive/functional.
O6	ETSI NFV: Ve-Vnfm-em	This interface is used for exchange between the entity managers of functions or applications and the corresponding EFS App/Function Manager, or between the EFS Service Platform entity manager and the corresponding EFS Service Platform Manager, and supports the following: • Application/function/service platform instantiation, update, termination, scaling in/out and up/down; • Application/function/service platform instance query to retrieve any run-time information; • Configuration of events and measurements regarding the application/function/service platform itself;
Om1	-	This interface is used for exchange between EFS App/Function Manager and EFS Service Platform Manager, and supports the following: • Notification of functions and applications information regarding instantiation, update, termination, migration, and scaling in/out and up/down with regards to their publication and subscription of services on the EFS Service Platform; • Notification of information regarding the availability of services at given locations; Compared to ETSI NFV environment, such interface is added to support the inter-dependency between functions, applications, and the service platform. Lifecycle management of functions and applications requires coordination with the EFS Service Platform as to satisfy the agreed SLAs.
Oo1	-	This interface is used for exchange between OCS Resource Orchestrator and OCS Stack Orchestrator, and supports the following: • Notification of events, measurement results, and usage records regarding edge/fog resources and EFS Stacks. Compared to ETSI NFV environment, such interface is added to support the dynamic environment where the mobility and volatility of resources are parameters that need to be considered for the lifecycle management of EFS Stacks.
T2	ETSI NFV: Os-Ma-nfvo	This interface is used for exchange between OSS/BSS and EFS Orchestrator, and supports the following:

ID	ETSI NFV/MEC ref. point	Description
		• EFS Stack Descriptor and EFS Stack lifecycle management, including EFS Stack instantiation, update, scaling, migration, termination, and query (e.g., retrieving summarised information about edge and fog resources associated to the EFS Stack instance); • Policy management and or enforcement for EFS Stack instances, function and application instances, and edge and fog resources (e.g., authorisation, access control, resource reservation, placement, allocation, etc.); • Forwarding of events, accounting and usage records and performance measurement results regarding EFS Stack instances, application and function instances, and edge/fog resources to OSS/BSS, as well as and information about the associations between those instances and edge/fog resources; • Integrating and releasing of resources into/from the target EFS including third-party information and SLAs. Compared to ETSI NFV environment, such interface also needs to support federation and privacy of multiple administrative domains coexisting in the same environment.
T4	ETSI MEC: Mm8	This interface is used for exchange between the Third-party(ies) Proxy and the OSS/BSS and supports the following: • Requesting the deployment of third-party applications/functions on the target EFS; • Integrating and releasing of resources into/from the target EFS including third-party information and SLAs. Compared to ETSI NFV environment, such interface is added to support interaction with third parties (instead of OSS/BSS) similar as done in ETSI MEC for the MEC application deployment.
T5	ETSI MEC: Mm9	This interface is used for exchange between Third-party(ies) Proxy and EFS Orchestrator and supports the same operations as T2. The difference resides in the two different endpoints: OSS/BSS in case of T2, Third-party(ies) Proxy in case of T5. Compared to ETSI NFV environment, such interface is added to support interaction with third parties (instead of OSS/BSS) similar as done in ETSI MEC for the lifecycle management of third-party applications.
T6	ETSI MEC: Mx1/Mx2	This interface is used for exchange between the OSS and the customer-facing service portal, and supports the following: • Requesting the deployment of third-party applications/functions on the target EFS; • Requesting the integrating and release of resources into/from the target EFS including third-party information and SLAs. Compared to ETSI NFV environment, such interface is added to support the deployment of third-party functions similar as done in ETSI MEC for third-

ID	ETSI NFV/MEC ref. point	Description
		party application. Moreover, this interfaces also supports the integration and release of third-party resources which was not considered in ETSI MEC.
E2	ETSI MEC: Mp1	This interface is used for exchange between the OCS and the EFS Service Platform and supports the following: • Forwarding of configuration information, failure events, measurement results, and usage records regarding edge and fog resources for monitoring purposes. Compared to ETSI NFV environment, such interface is added to support the publication and consumption of data from and to the service platform. This is similar to what is done in ETSI MEC for service registration, service discovery, and communication support for services. Nevertheless, this interface also supports services to be published and/or consumed by physical/virtual resources, functions, and OCS components in addition to the sole applications.
F2	ETSI NFV: Or-Or, Vi-Vi	This interface is used for exchange between different OCSs when federation at OCS level is in place. Such interface supports the functionalities defined for O2, O3, O4, Om1, and Oo1 interfaces. Federation agreements may limit the functionalities over this interface, either in availability or scope. Compared to ETSI NFV environment, such interface also supports the interaction between EFS Managers in the scope of EFS Service Platform.

3 OCS Design

This section introduces the OCS design choices aimed at fulfilling the OCS requirements introduced in Section 2.2. Particular focus is given to addressing the peculiarity of 5G-Coral environment, such as heterogeneity of resources, mobility and volatility of resources, 5G-Coral-specific monitoring information and system, and the necessity of supporting third-parties. Ultimately, 5G-Coral architecture is tightly coupled with prominent paradigms of next generation mobile communications, i.e., ETSI NFV and ETSI MEC. Although these standards already cover key building blocks of future 5G architecture, such as virtualisation and edge computing, we identified a set of additional requirements that need to be fulfilled, ranging from supporting a wide variety of heterogeneous resources to handling dynamic and volatile resources and enabling migration and monitoring of such resources. In the following, we will extensively cover each of these aspects and motivate the design choices we made, with the intent of bridging the gaps between current state-of-the-art system architectures and the above requirements.

3.1 Heterogeneous resources support

Motivated by the synergy of edge and fog in bringing networking and computing capabilities closer to the users, the OCS needs to consider the wide variety of computing devices, including volatile and non-stationary, to achieve an integrated edge and fog system. To address such heterogeneity, the OCS needs to support **multiple virtualisation subsystems** specific to each category of devices. For instance, hypervisors like KVM[1] and XEN[2] are very well-suited to edge environments where powerful servers are likely to be present. However, those supervisors have a computational requirement that is too demanding for resource-constrained devices. To this end, new virtualisation technologies are arising, such as the Intel ACRN[3]. This new hypervisor enables to run different software subsystems (VMs) on a system-on-chip (SoC) instead of requiring high-level data-centre CPUs. Moreover, ACRN embeds an additional layer of security with very limited performance impact that allows to run safety-critical VMs on a protect portion of the SoC and less critical VMs to run in the normal portion of the system. One use case for this hypervisor can be in automotive, in which it is possible to have the dashboard for the driver that runs on the secure portion, and then the infotainment that runs on the normal portion of the system.

To cope with such heterogeneity of devices and virtualisation subsystems, the OCS needs to adopt an architecture that is easy to extend while being lightweight on the target resources. To address this, 5G-Coral adopts a **plugin-based architecture** for the OCS as to allow a quick integration at architectural level of any new hypervisors, operating systems, CPU architecture, etc. Furthermore, the OCS needs to manage distinct types of applications and services that can be deployed in the EFS. These applications and services may be composed by heterogeneous instantiable software components, having this possibility leads to the need of supporting different virtualisation technologies such as VMs, Containers, Unikernels, bare metal and so on, from and OCS point of view this means supporting different hypervisors for each of these categories of instantiable software. Finally, the OCS network management needs to adopt the same plugin-based architecture as to enable the interaction with different network technologies (e.g., virtual switches, SDN controllers, etc.) and the chaining of complex network services (i.e., EFS Stack) across multiple environments.

An **EFS Stack** can be viewed architecturally as a forwarding graph of functions and/or applications interconnected by supporting edge and fog resources and/or service platforms. A specific requirement in edge and fog environment is related to the proper description of **I/O interface.** This is particularly relevant since functions and applications may need access to low-level details of network interfaces (e.g., Wi-Fi, 5G NR) to perform networking tasks (e.g., BBU, Wi-Fi Access Point, etc.). Another requirement comes from the volatility and mobility nature of resources that may cause intermittent connection or, in the worst case, a total disconnection of the resources from

[1] http://www.linux-magazine.com/Issues/2008/86/KVM
[2] https://xenproject.org/developers/teams/hypervisor.html
[3] https://projectacrn.org/

the OCS. This means that the descriptor should consider the different volatility and mobility behaviours as to allow the OCS to react accordingly to the foreseen situations. For instance, the OCS may configure backup links or decide to place some critical components of the application or function to a resource with more stable connection if possible. Besides, hardware **accelerators** need to be considered in the descriptor as to allow applications and functions to offload heavy computation, such as GPU-based image recognition or hardware-based encryption. Descriptor should also support privacy-related aspects, including the possibility of defining geo-fencing, aspect particularly relevant in case of moving resources. Finally, additional information on the target execution environment of function and application needs to be provided in the EFS Stack descriptor. This is to allow to select the right resource for instantiating or migrating the involved atomic entities. For instance, it is necessary to specify the specific hypervisor technology (e.g., VM or container) required to on-board and run the application or function.

ETSI NFV defines descriptors for both Virtual Network Functions (VNFD) and Network Services (NSD) [1] which encompass deployment needs that applications may require. An NSD is defined as a chain of VNFs and Physical Network Functions (PNF) with their corresponding descriptors, namely VNFD and PNFD, respectively. The NSD defines as well one or more Service Access Point (SAPs) to access the deployed service, and a set of virtual links interconnecting the network functions along with the QoS requirements (see Figure 3-1).

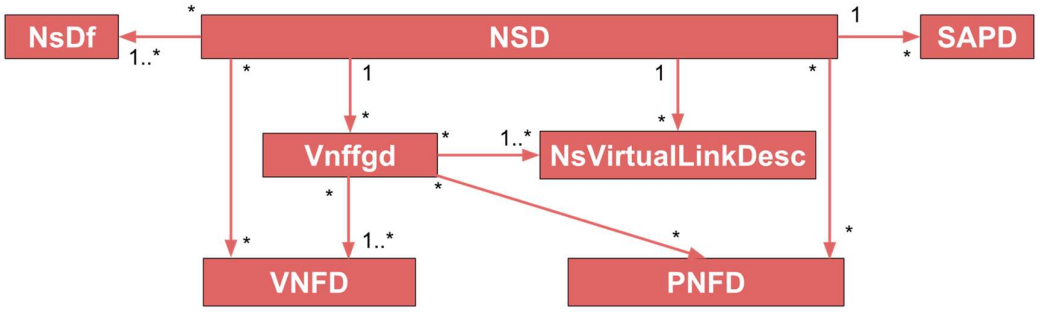

FIGURE 3-1: ETSI NETWORK SERVICE DESCRIPTOR UML

Since an NSD might vary in terms of number of instances, affinity or even amount of resources required; ETSI NSD and VNFDs [2] have Profiles and deployment flavours. A NsProfile and VNF profile is nothing but a specification of how many instances and a set of affinity details related to the NS or VNF they are defining. Both of them refer, respectively, to NS deployment flavours (NsDfs) and VNF deployment flavours (VnfDfs) that provide scaling and life cycle management information. In the particular case of a VnfDf, the object references the set of Virtual Deployment Unit (VDU) profiles (VduProfile) for affinity and monitoring purposes of the related VnfDf, as well as a list of actual VDUs needed for the deployment of the VNF associated to the deployment flavour. Such VDU elements specify Virtual Link Descriptors (VLD), VDU connection point descriptors (VduCpds), VirtualStorageDesc, and VirtualComputeDesc, wherein CPU and memory details are defined.

TABLE 3-1: KEY FIELDS IN ETSI AND TOSCA DESCRIPTORS

Scope	Field	Supported in ETSI NFV	Supported in TOSCA
NSD	Redundant topologies	Yes	Yes
	Handle network failures	Yes	Yes
	Migration	No	Yes
	Region/domain deployments	Yes	Yes
VNFD	Protocol specification	Yes	Yes
	Direct access to interfaces	Yes	Yes
	Portability	Yes	Yes
	Lifecycle management	Yes	Yes

The **EFS Stack descriptor** is based on ETSI NFV NSDs, ETSI MEC Application Descriptors and the migration specifications present in TOSCA [32]. TOSCA is a language used to describe network services to be deployed. Its latest version follows the ETSI NSD specification, and extends it including information related to migration that is not present in ETSI NSD. Redundant topologies to allow pre-provisioning, life cycle and network failure management is covered in ETSI NSD; while dealing with services migration is not considered in the specification. Although TOSCA descriptors are an implementation that follows the ETSI NSD specification, they consider migration aspects. We incorporate this key parameter in our EFS stack descriptors for the 5G-Coral, see Table 3-1. TOSCA description language is based on YAML[4] and supports the existence of multiple tenants in different domains. Like the ETSI NSD/VNFD, it can describe dependencies between services, requirements, and capabilities of each VNF to be deployed; hence it implements the standard requirements. Edge-related parameters are not considered in TOSCA, and things like devices volatility cannot be modelled with the language. This is because, as said before, it is based on ETSI NSD which does not consider the fog environment that 5G-Coral addresses. Because of these shortcomings, we decided not to use TOSCA and rather define our services with the EFS stack descriptors.

TABLE 3-2: NOVEL PARAMETERS PRESENT IN EFS STACK DESCRIPTORS

Parameter	Description
I/O port constraints	Describes port-specific requirements for the device hosting the service.
Location constraints	Location coordinate and radius where a VNF should be deployed ([1] considers the location of a PNF with the geographicalLocationInfo).
Interface types	This parameter specifies different internet technologies that can be requested in the deployment as CAN-BUS or Wi-Fi.
Volatility information	Expresses for how long devices in the fog are up and running to host services.
Low level network specs	This parameter specifies additional parameters specific to the network interface. E.g., wireless channel, modulation, DHCP range, multicast groups, etc.
Migration triggers	Specify a set of values that might trigger the migration of a service. These values are related with the underperformance of the service.

EFS Stack descriptor addresses the heterogeneity present in 5G-Coral by defining a set of new fields for the EFS Stack that are not considered in the ETSI NFV descriptors (see Table 3-1). It is true that ETSI allows to specify the location of a PNF to be used within a service (geographicalLocationInfo parameter), but it is not possible to impose a location constraint for the VNFs within the NSD. Our descriptors can specify where every component of a NS has to be deployed, hence we have a full geolocation control in 5G-Coral. With the port and interface parameters, our descriptors can ensure that services needing a Wi-Fi interface are deployed in devices that actually have a Wi-Fi interface. Other fog specific issues such as devices volatility are addressed with our novel descriptors thanks to the specification of the minimum amount of time we impose for our service to be running. And last but not less important is the migration issue. In the fog, devices can power off leading to a service deletion, and in such case, it is needed to perform a migration. Our descriptors cover this situation as well as underperformance triggers that might lead to migrations.

When deploying an application in the 5G-Coral, it is necessary to specify how the instantiated application will interact with other ones already deployed providing services, or even how it is going to expose its services. EFS stack descriptors incorporate ETSI MEC [39][40][41] application descriptors information. These descriptors can provide information related to DNS resolution to deployed services in 5G-Coral, which protocols are used to access the service given by the application, and even how it serialises the information given by that service (JSON, XML or PROTOBUF3). Another thing we incorporate in the EFS Stack descriptors is the interaction and

[4] YAML is the dominant Data Language Specification (DLS) for modelling the deployment and orchestration of cloud applications

dependencies between applications and services that may be present in outside the deployed EFS Stack (see Figure 3-2). If an instance application uses another one (for example a location service), our descriptors specify the protocol used to access it, the serialisation format, and permission-related information.

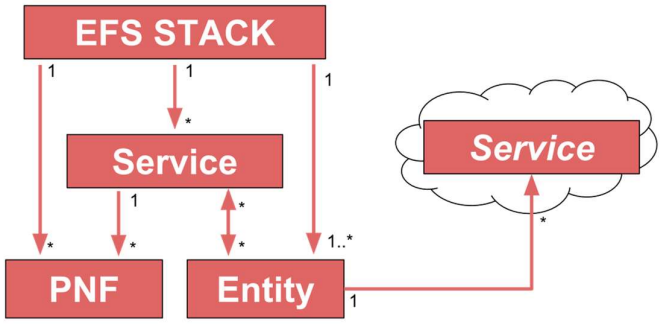

FIGURE 3-2 EFS STACK DESCRIPTOR INFORMATION MODEL

An EFS Stack descriptor is composed of entities and PNFs that can provide and consume services. An **entity** (see Figure 3-3) is envisioned as a compound of atomic entities, which are components with a specific functionality; for example, an atomic entity can be a database, and an entity can be a web service formed of a load balancer, multiple web servers, and a database. These enumerated items would be atomic entities connected among them in a network, and they can impose deployment constraints and define connection points that will be used to access them.

Hence a **service** can make use of these entities to perform the operations needed to provide information to other entities that might consume it. For example, a weather report service can make use of multiple entities that collect temperature, humidity and wind information; then that information is serialised and exposed to another entity consuming such service.

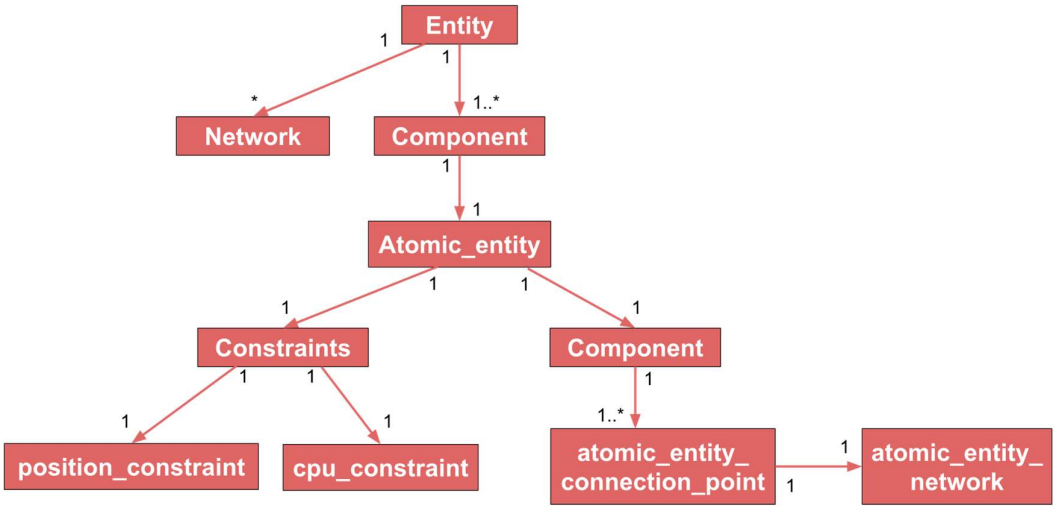

FIGURE 3-3: EFS ENTITY DESCRIPTOR INFORMATION MODEL

3.2 Dynamic resources support

Dynamic deployment of elements of EFS and OCS is pivotal for the functioning of 5G-CORAL system. Discovery, integration and careful selection of computing resources with respect to their availability and capability is critical for optimised software element migration and placement. This task can be even more challenging in dynamic environments containing volatile and unpredictable

fog resources. In this section various aspects of interaction between OCS and dynamic resources are discussed.

One of the key requirements of the OCS is to harvest computational power of low end and often volatile resources abundance of which is located at the edge. The most notable advantage coming from using these devices is the vicinity to the end user which translates directly into a very low latency but also (indirectly) to the autonomy in operation. Systems comprised of edge devices residing in the local network, supported by local OCS, can be fully operable without the need to keep connection with the cloud at all times. These advantages make these resources, when managed and orchestrated in the right way, a base for new applications and services.

In order to utilise the potential of low end and volatile resources, first these resources need to be integrated into the 5G-CORAL system by means of EFS and OCS embedded mechanisms. More details about the integration process can be found in Section 4.4. Afterwards, the EFS Resource Orchestrator may allocate these resources for the EFS applications and functions to meet the requirements of NFV service. In order to do so, a careful examination of the resource status is required. Indeed, volatility of the resources may not meet application requirements towards availability of the services it provides. The functioning of critical applications such as collision avoidance or rapid medical response might be jeopardised by the computing resources being shut down unexpectedly or just moved away from the network. To prevent such course of action, techniques such as monitoring and prediction of the state of the resource would have to be employed by the OCS. By knowing the availability pattern of the resource or the trajectory of movement resources can be used safely while minimizing the down times of the applications and functions running on top.

OCS elements, described in Section 2.3, are responsible for executing NFV services through the interaction with each other and with other EFS systems. Traditionally placed in the cloud, the management and orchestration system of computing resources is now perceived to be more dynamically distributed between cloud and edge continuum. The reason for that is twofold. On the one hand, the latency constraints of the EFS system elements require the decision-making entity to be placed near or at the edge. On the other hand, an isolated system, due to the mobility, security concerns or simply poor WAN connectivity, may not afford to rely on the OCS being placed entirely in the cloud. As a result, OCS system components might be deployed on EFS resources in an analogous way as EFS applications and functions. Such deployment is a trade-off between mentioned environmental requirements and volatility of local resources.

OCS distribution, allowing OCS elements to be placed in different physical, interconnected nodes constitutes a major challenge due to the significant design shift from centralised orchestration and management system view of ETSI NFV/MEC architectures. Volatile and mobile resources bring some degree of unpredictability that is related to their availability. Unpredictable behaviour of the device can create an error-prone environment which in turn brings concerns related to the service availability of the software deployed on top. An OCS as distributed structure may suffer from this issue as well. To ensure service availability of the OCS, one solution could be to request resources to provide reliability certification (such as ISO 9001) to give information about failure probability. This, however, says little about the way the device operates (e.g., frequency and duration of idle times). Therefore, in this project, this problem is approached from another angle. We recognise the importance of the monitoring of the resource behaviour to detect availability and mobility patterns and, in result, classification of the resources regarding stability of operation. An OCS equipped with this knowledge could safely migrate OCS components among different resources in such a way that minimises the downtime of a given function. On the other hand, if the failure of one or few nodes happens, a self-healing mechanism can be employed to recover from the damage. In this case techniques such as forward or backward error recovery can be used to cope with hardware faults [47].

The OCS will be distributed across different nodes, also in terms of network connectivity. This can lead to have part of the OCS running on devices with volatile networks connections because, as said, part of it can run on relatively constrained devices. It is important to leverage on a communication protocol that allows to not be aware of the intermittent connectivity, to make this

possible part of the OCS can leverage the abstraction provided by a middleware to make the control plane not be destroyed in case of network disconnections and reconfigurations, and be able to take the right actions to rearrange network configuration for VNF. To address these problems, the OCS leverages on the abstraction of a distributed key-value (K,V) store built on top of a communication middleware able to not be destroyed by poor or intermittent connectivity as well as to scale to a large system. Using this data centric approach on the management plane allows the OCS to communicate about the *what* instead of the *where*. Such data-centric pub/sub approach to the control/management plane allows to abstract the underlying communication technology used, thus allowing to have parts of the OCS running on very different nodes, both in terms of computing and networking capabilities. This enables the OCS to be independent on where a component is placed, and whether it is reachable or not, because the information on the control/management plane is eventually consistent thanks to the middleware abstraction, thus enabling the system to recover from some node disconnection.

Supporting a dynamic environment integrating resources often belonging to different owners, having a volatile nature, but also spread across many networks, requires a comprehensive discovery approach. The objective of the discovery as such is to attach new EFS resources to the domain and hence make them usable by the domain's OCS. The discovery procedure shall be carried out in the plug-and-play fashion regardless of the initial configuration of the computing resource. The result of the discovery process is a particular knowledge about the resource. Only then a resource can be considered discovered. This knowledge includes several aspects:

- Hardware (CPU, RAM, storage, etc.);
- Software (OS, protocol suite, etc.);
- Networking services (routing, security, etc.);
- Context (location, ownership, etc.);
- Topology (interconnection with other network entities).

What differs from the typical discoverable items is a context information. Indeed, in order to effectively include the device into a virtualised infrastructure, the OCS should not only know its capability, but also information about location, mobility, availability or ownership. While some context information, like mobility and availability, might be helpful to judge resource stability against application QoS requirements, others, like ownership may solve out security, charging and thus integration issues.

It is desired that discovery is performed in a dynamic manner. Therefore, it is assumed the discovery should not rely on preconfiguration of the newly added EFS resources with information of how to directly assess local OCS through the EFS. The information should be found out dynamically through the discovery process, which may include:

- Network bootstrapping – device must discover its neighbours and join the network:
    - Medium access (Wi-Fi, Ethernet, etc.) – direct communication with the neighbours (computing resources, networking elements), in addition to physical discovery of the neighbour presence (e.g., scanning Wi-Fi beacons) it includes establishing means of communication with the neighbour (e.g., discovering its IP).
    - Participation in the network – node may need to become part of the topology. In order to do so, it must run topology related protocols e.g. to create mesh topology with other nodes.
- Service discovery – device needs to exchange services/information that is related to its capabilities or state. Alternatively, it needs to discover credentials of the EFS (IP, port number, etc.) to establish 1-to-1 connectivity with the OCS for further information exchange. Service discovery can be achieved through several application level protocols:
    - Name advertisement – searching for the device (IP address) through the name (DNS, mDNS, LLMNR, etc.)
    - Service advertisement – searching for the service (SSDP, DDS, SLP, DNS-SD, WS-Discovery, etc.)

In order to maximise support for the volatile resources, 5G-CORAL project aims at converging these traditional discovery steps into one happening at the network accessing phase. In addition to network related information passed to the device via access protocols (e.g., 802.11), a device may be informed about the number and characteristics of local EFSes. The process of discovery for different access technologies is described in Section 4.3.

3.3 Dynamic migration support

Dynamic response to the ever-changing environment requires OCS to perform ad-hoc migration of applications and functions and is a part of dynamic resource management. The process of moving software from one EFS resource to another must take into consideration the resource status (availability, capability) and based on that virtualisation technology. We distinguish VM and container migration. Both methods have their advantages. In subsequent sections we will explore their applicability for dynamic resources.

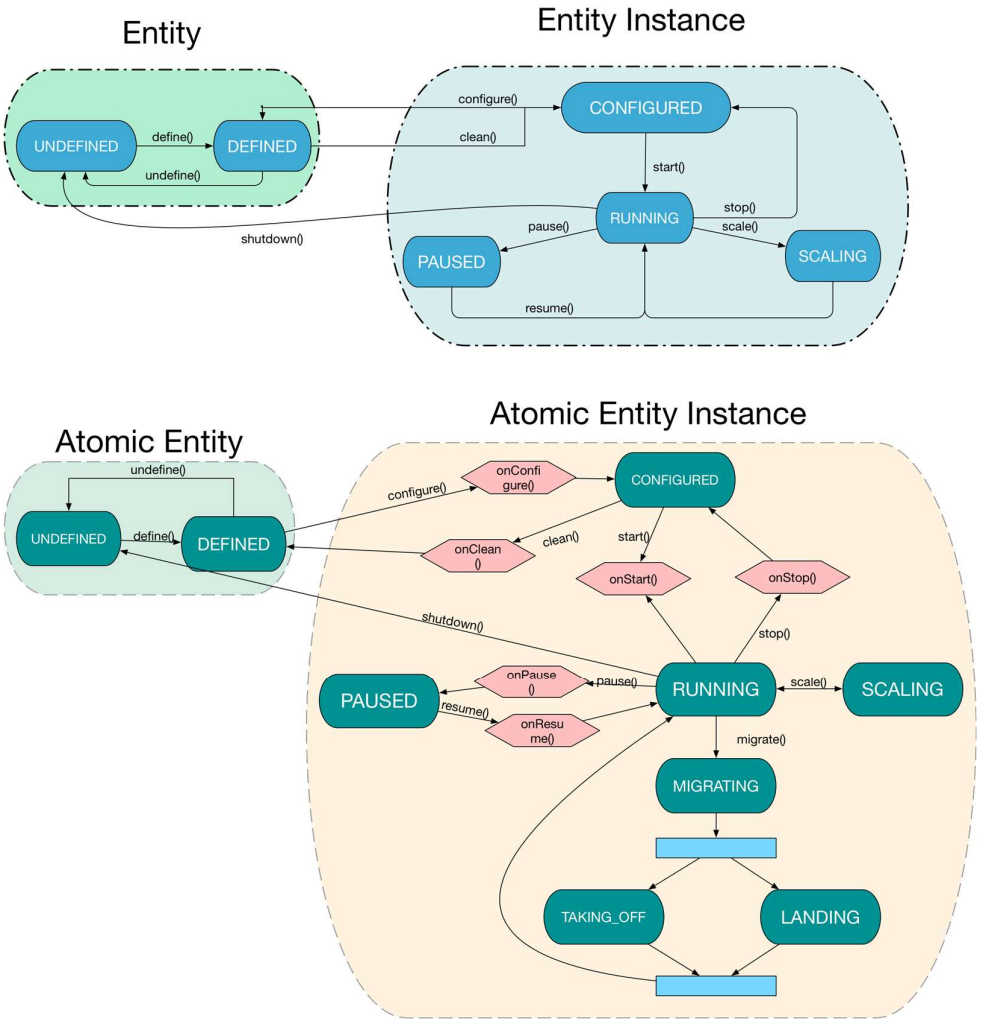

FIGURE 3-4: FSM FOR ENTITIES AND ATOMIC ENTITIES IN OCS

For each of the components defined in Section 3.1, a Finite State Machine (FSM) that describes the lifecycle of NSs and VNFs is also defined. These state machines are generic because they need to represent heterogenous types of instantiable software. This generalness lead to the possibility some of the state cannot be reached for some types of deployable units, for instance it is not possible to live migrate a non-collaborative native application. In the FSMs there is also a

separation between the states that are represented by descriptors and the ones that are represented by records; the first ones are the templates for NS and VNF, while the other ones are the actual VNF and NS instantiated. In Figure 3-3 it is possible to see two different state machines. The first one represents the possible states of an Entity, while the second one the possible states of an Atomic Entity. The second one contains more states than the first one because of the fact that an atomic entity is an indivisible unit of deployment.

The FSM for atomic entities describes also the state transitions that will be implemented for each type of atomic entity, this means that to add the support for a new type of atomic entity it is necessary to implement a plugin allowing these state transitions and the relative call-backs. As said, this allows having an easily extendable architecture and the possibility to support new types of deployable units. Table 3-3 and Table 3-4 describe in detail the states starting from the ones that can be assumed by the entities. Some state follow a different convention of the one used by ETSI NFV because of the fact that we have ad high distributed environment, this is clear in the case of the DEFINED state that can be easily mapped to the ONBOARDED state from ETSI NFV, but in our case the name is different because all the system knows information about that entity/atomic entity, but only one node contains the actual package. Also UPDATE/UPGRADE transition are masked by the fact that in our information model each entity/atomic entity has a version number, this means that an update is mapped to a new definition with updated fields.

TABLE 3-3: ENTITY STATES DESCRIPTION

State	Call-backs	Description
UNDEFINED	• undefine()	This state means that the entity is not yet defined in the 5G-CORAL architecture, so is literally a non-existing state, this state can be reached using the undefine() method from an existing Entity.
DEFINED	• define() • clean()	An Entity in this state is described by a NSD so is present in the set of NSs of the 5G-CORAL architecture, it is possible to instantiate this service.
CONFIGURED	• configure() • stop()	In this state each of the component of the NS is in the *CONFIGURED* state, means that disks images, networks, and the NS can be started.
RUNNING	• start() • resume()	The network service is up and running, each of the component is in an operative state.
PAUSED	• pause()	The NS is paused, all the components are not running but are not destroyed, this should maintain the state for continue the execution.
SCALING	• scale()	The NS is scaling (up/down) each of the component is scaling, this can also mean that we are replicating some component or destroying some replicas.

TABLE 3-4: ATOMIC ENTITY STATE DESCRIPTION

State	Call-backs	Description
UNDEFINED	• undefine()	This state means that the atomic entity is not present in the 5G-CORAL architecture.
DEFINED	• define() • clean()	In this state the VNFD of the atomic entity is stored in the 5G-CORAL system it is possible to instantiate that VNF, also the base disk image is retrieved and stored into 5G-CORAL.
CONFIGURED	• configure() • stop()	The atomic entity is instantiated, the disks, networks, configuration files and HV-specific descriptors are created and registered, in the case of KVM/XEN/OpenStack/LXD the domain is created but is not yet running.
RUNNING	• start()	The atomic entity is up and running.

	• resume()	
PAUSED	• pause()	The atomic entity is paused, this means that the execution is interrupted, but the state is preserved for resuming.
SCALING	• scale()	Some of the compute/storage/network requirement of the VNF are scaling up/down.
MIGRATING	• migrating()	The atomic entity will migrate soon, this can lead to two different states, *TAKING_OFF* in the source node and *LANDING* in the destination node.
TAKING_OFF	N/A	The atomic entity (and its flavour and base image) will be defined in the destination node, in case of live migration we leverage the underlying hypervisor to perform the migration, otherwise we will destroy the instance in this node and create on the other. This state lead to *DEFINED*, because in the source node the instance of the atomic entity will be destroyed.
LANDING	N/A	The atomic entity (and its flavour and base image) will be defined in this node, and then in case of live migration the underlying hypervisor will perform the actual migration, otherwise we will stop on source node and start on the destination node, this state leads to *RUNNING*, because in the destination node the VNF will be in running state.

3.3.1 Migration of EFS entities in a VM based environment

Hypervisor-based virtual machine (VM) allows the encapsulation of a guest kernels, a guest operating system and applications in an instance running on top of a host kernel. This encapsulation ensures isolation and provides resource allocation and provisioning for multi-tenant environments such as cloudlets. VM migration is a mature technology currently being used in commercial cloud solutions such as vMotion of VMware. 5G-CORAL supports VM live migration to provide uninterrupted service delivery and to maintain the latency and bandwidth requirements by functions and applications. Pre-copy migration scheme is one of the supported live migration schemes in which iterative dump is used to reduce the downtime during the migration process.

VM migration in the context of 5G-CORAL differs from that of cloud environment. That is, functions and applications are migrated over wide area networks (WANs) which is how MECs are envisioned to be interconnected. In data centres, VMs are usually built on top of shared storage and network enabled storage such as storage area network (SAN). In this case, VM static files (disk storage) are not moved to the destination host and only in-memory data, which include kernel state such as TCP control blocks for active connections, and application-level state, is migrated.

3.3.2 Migration of EFS entities in a container-based environment

Hypervisor-based virtualisation offers a variety of platform selection and well-defined isolation between virtual instances and host system. However, VMs incur large overhead in terms of storage and computing capability. Each running VM emulates a guest kernel and a guest OS for application and function to be implemented. This involves a very inefficient use of resources, especially when the size of the deployed application is very small when compared to the footprint overhead of the kernel and OS. To address this, 5G-CORAL also supports light virtualisation technology such as containers. Linux Containers (LXC) is an operating-system-level virtualisation which allows the existence of multiple isolated instances running on top of the same kernel. This container virtualisation technology removes the overhead incurred by the guest kernel in terms of storage, CPU time and hardware emulation. For this reason, containers are more suitable for less capable edge platforms.

Container migration is a new area which is currently being investigated. 5G-CORAL provides a basic support for container migration which can be utilised to live migration containerised functions

and application with minimal downtime. Mechanisms for pre-copy container migration will be supported and detailed in deliverable D3.2.

3.4 Monitoring support

Monitoring is an essential part of the OCS platform, which is in charge of controlling compliance with SLAs and all requirements the OCS platform must achieve. Moreover, due to the dynamicity and volatility of resources in the edge and fog system, the OCS is required to be capable of reconfiguring networking resources when devices appear or disappear, this can only be achieved by controlling every aspect inside the EFS and the underlying infrastructure.

To monitor the system, it is necessary to have an agent in charge of collecting the data and another one being executed on the devices to send the monitoring information. In 5G-CORAL, an EFS Monitoring Service will be in charge of collecting published data from monitored devices and serving it to the OCS.

Monitoring is a task of paramount importance for both providers and consumers. On the one side, it is a key tool for controlling and managing hardware and software infrastructures; on the other side, it provides information and Key Performance Indicators (KPIs) for both platforms and applications. The continuous monitoring of the architecture and of its SLAs (for example, in terms of availability, delay, etc.) supplies both the providers and the consumers with information such as the workload generated by the latter and the performance and QoS offered through the architecture, also allowing to implement mechanisms to prevent or recover violations of agreed terms (for both the provider and consumers). Monitoring is a key factor in terms of orchestration, providing inputs for the mechanism being performed by the orchestrator.

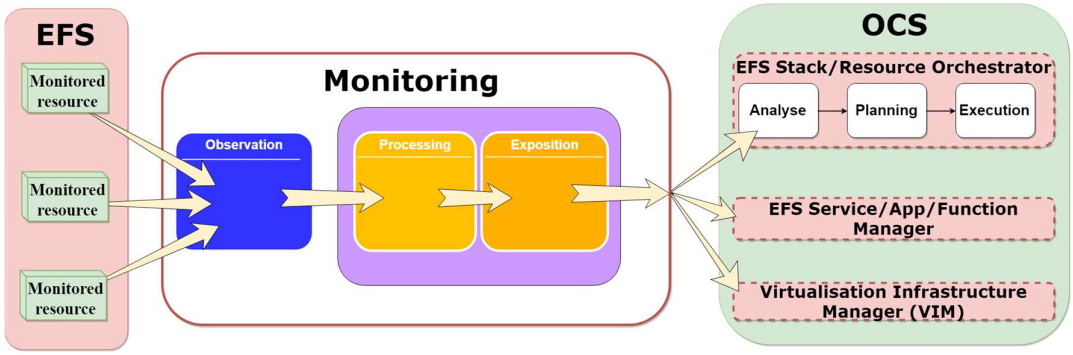

FIGURE 3-5: MONITORING SERVICE AND FUNCTIONAL DECOMPOSITION

A monitoring service should fulfil distinct functions: observation of the monitored resources, data processing, and data exposition, as depicted in Figure 3-5. The processing function may be further detailed into sub-operations such as measurement aggregation (e.g., to adjust sampling), transformation of measurement into events, event processing and notification management. To deploy such a monitoring service on a Fog/Edge-based operator infrastructure with a high availability, the monitoring service must satisfy the properties shown in Table 3-5.

TABLE 3-5: MONITORING PROPERTIES

Property	Description
Scalability	The monitoring service should be able to supervise a large number of resources. Moreover, it should handle a sudden growth of load. That is, it should adapt to an increase of request rate from the different management systems and to an increase or a decrease of the monitored resources without any impact in terms of performance. The measurement sampling may also have an impact on processing and network capacity.
Resilience to resource volatility	The monitoring service cannot prevent the failure of the resources hosting it. It should be designed to allow its adaptation to any resource removal. That is

	particularly important at the edge of the target infrastructure or in the context of virtualised environments where the resources are less reliable.
Resilience to connectivity failures	The massively distributed fog/edge-based operator infrastructure is vulnerable to network failures particularly at the edge-devices level. Moreover, the monitoring service heavily relies on the network to observe remote resources. For instance, it should ensure the retransmission of data lost in case of network failures.
Modularity	The heterogeneous resources of the Fog/Edge-based operator infrastructure range from high performance servers to modest performance user terminals. To offer the monitoring service more choices of deployments, its deployment should be made possible on any type of these resources regardless of their capacities.
Locality	The monitoring service should ensure adequate response delay regardless of the location of the monitored resources which is particularly challenging in the case of massively distributed infrastructures. It may be required to deploy the monitoring service nearest to its users, such as centralised management systems, as well as to the monitored resources that may be localised at the infrastructure edge level.
Multi-tenancy	The monitoring service should be able to provide measurements per tenant-level as well as to translate monitoring metrics from virtual-aggregated level into the physical parameters and vice-versa. This monitoring property is required for ensuring the proper isolation between tenants in terms of performance and SLAs.

The 5G-CORAL environment envisages the cooperation of a variegated set of resources, devices, connectivity technologies, etc. that need be monitored either actively, periodically, or on-demand. While a considerable large amount of monitoring platforms is available in literature (see Appendix B for a review of state of the art), none of them is suitable for an edge and fog environment where resources can be mobile and appear and disappear anytime. For instance, cloud monitoring tools usually require constant connectivity, huge bandwidth for continuous monitoring, and often present a considerable footprint on resources consumption, thus being not suitable for resource-constrained devices. To this end, we hence decide to monitor the resources via a service offered by the EFS Service Platform. Such platform is meant to serve as message delivery platform in environments where constant connectivity is not assumed. Each EFS resource composing the EFS-VI is hence the publisher of monitoring data that can be then consumed by the OCS. To save bandwidth, we do not require continuous monitoring in the sense that resources can intermittently report their data. Moreover, reporting events can be configured by the OCS to be triggered by a timeout or data exceeding a given threshold. Therefore, in the following, we identify a set of monitoring information that needs to be communicated to the OCS. Table 3-6 reports the measurements and information required to be supported by the OCS and EFS resources.

TABLE 3-6: MONITORING MEASUREMENTS

Measurement	Description	Granularity
Network bandwidth	Network bandwidth experienced by resources can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the measured bandwidth is below a given threshold (Mb/s).	Measurement granularity may depend on the mobility of the resources which may result in higher variability of the available bandwidth, thus potentially requiring
Network latency and jitter	Network latency and jitter experienced by resources can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the measured latency or jitter is above a given threshold (ms).	

Network packet loss rate	Packet loss rate experienced by resources can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the measured packet loss rate is above a given threshold (%).	more frequent measurements. The measurement end-points may change depending on the SLA for the resource, the hosted applications, functions, or published/consumed services.
Network parameters and information	Specific network parameters and information need to be measured on-demand or periodically. Such parameters and information are specific for each access technology. E.g., configured channel, transmission power, employed modulation coding scheme, MIMO, channel SNR, etc.	
Storage sequential I/O bandwidth	Sequential read and write bandwidth on the storage available locally on the resources (Mb/s).	Measurement usually performed when integrating the resource into the EFS as to access storage performance.
Storage random I/O bandwidth	Random read and write bandwidth on the storage available locally on the resources (Mb/s).	
Storage space availability	Available storage on the resources for hosting applications and functions which can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the available storage is below a given threshold (MB).	Measurement required when instantiating an application/ function and during execution tie as to ensure that storage is not over-used.
CPU utilisation	Overall CPU utilisation of the resources including per-application and per-function CPU utilisation profile. Such parameter can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the overall resource CPU utilisation, or the CPU utilisation of a given application or function, is above a given threshold (%).	Measurement required for continuous monitoring of the status of the resources as to avoid resource overloading and to eventually trigger migration or scaling of applications and functions. Periodic measurement at second granularity.
RAM utilisation	Overall RAM utilisation of the resources including per-application and per-function RAM utilisation profile. Such parameter can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the overall resource RAM utilisation, or the RAM utilisation of a given application or function, is above a given threshold (MB).	
Power mode	Power mode of resources (e.g., idle mode, sleep mode, active, etc.), including battery level in case of battery-powered devices. Such parameter can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the battery capacity is below a given threshold (%).	Periodic measurement at minute granularity.
Location	In case of resources are equipped with location sensors (e.g., GPS), information about device location can be directly provided by the resource itself. Such parameter can be measured on-demand or periodically. In case of periodic measurement, an alarm could be configured and fired when the resources leave or enter a given area (GPS coordinate).	Periodic measurement at second or minute granularity depending on the mobility of the resource.

Billing/ Accounting	In case of federated resources or third-party applications/functions deployed on the EFS, monitoring of the utilised resources is required for a proper billing.	Periodic or on-demand measurement at the granularity defined in the billing system.

3.5 Third-parties support

The 5G-Coral environment is characterised by multiple administrative domains coexisting in the same service offer area that are required to cooperate with the ultimate goal of offering an integrated edge-fog system. Specifically, an **administrative domain** is defined as a collection of resources operated by a single organisation, and it is viewed as a cohesive entity and its internal structure is unimportant from the outside. Domain's resources are assumed to interoperate with a significant degree of mutual trust among themselves but to interoperate with other administrative domains in a mutually suspicious manner. The external administrative domains are hereinafter also referred to as third-parties. To achieve the envisioned cooperation between multiple domains, a federation mechanism is required to achieve the desired level of integration. Notably, **federation** is a mechanism for integrating multiple administrative domains at different granularity into a unified open platform where the federated resources can trust each other at a certain degree. Whereas **trust** in federated domains is the embodiment of a service/business-level agreement or partnership between two organisations. Figure 3-6 shows the approach followed in 5G-Coral for federation where the classical approach of loosely and statically federating/integrating domains is tightened and endowed with flexibility.

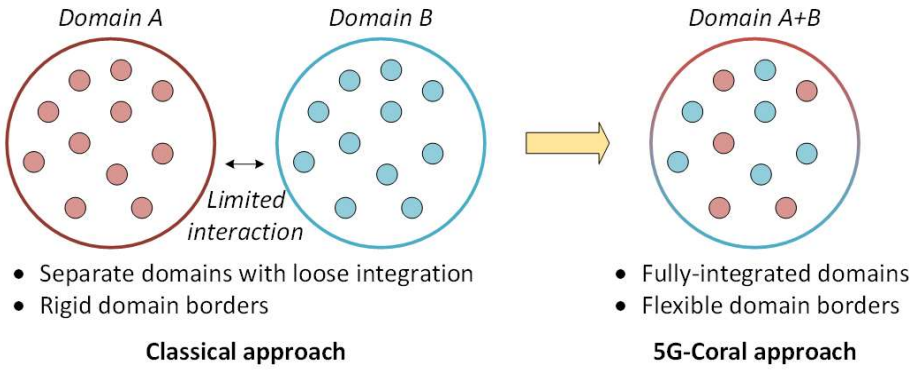

FIGURE 3-6: 5G-CORAL FEDERATION APPROACH

Historically, only a limited set of system capabilities have been considered for federation, thus keeping the involved domains well separated and loosely coupled. Interaction between domains occurred only amidst the domain borders and resources in each domain had very little interaction with external resources. This approach is well-suited in cloud environments where the resources of each domain are physically deployed in few well-identified locations. For example, federated identity systems (e.g., single sign-on) link a user's attributes to multiple systems. This allows a user to authenticate against multiple systems but not necessarily to be capable of using them with the same level of authorisation, given indeed the distinct degree of trust that could be in place between the different administrative domains. Given the peculiarity of the 5G-Coral environment, where variegate resources coexist in the same service area, the hard-border between these domains necessitates to blur out for achieving a real integration in the edge and in the fog. That is, resources are expected to cooperate locally with external resources. Finally, federation is also used to manage **pricing** in service industries where the necessity of bundling services and invoice customers syndicates their service definitions and pricing determination. Such flexible definition of services and pricing allows the introduction of new pricing models in market-oriented time. For example, resource sharing between different EFSs may involve payments between the operating administrative domains.

5G-Coral considers two types of federation: static (off-line) and dynamic (on-demand). In static federation, the interested administrative domains pre-establish federation relationship before EFS operations. In dynamic EFS federation, federation relationship is established on-line and in an on-demand manner. We call it an open federation. Each administrative domain publishes an abstraction of its available resources to peer domains. Other domains may subscribe to this publication and submit their resource requests when needed. The federation relationship starts from the lease of the resource and ends at the return of the requested resource. To pave the path towards a 5G-Coral federation, we extend the definition of resources in [34] by applying the concept of trustiness.

We define as **trusted resources** those resources that are owned by the same entity owning and operating the infrastructure. This allows defining the mutual trust among trusted resources as the capability of directly communicating at layer 2. That is, resources are capable of accessing the EFS and communicating with the OCS because they are trusted meaning that they can be authenticated and authorised by the target system. In this case, security is enforced either during physical deployment or by providing credentials beforehand. Similarly, we define as **untrusted resources** those resources that are owned by a different entity owning and operating the infrastructure. Untrusted resources need to grant layer 2 connectivity in the first place in order to connect to the network and communicate with the target EFS and OCS. The 5G-Coral architecture defines the interface T6 to support third-party organisations to integrate their resources into the EFS. The interface directly connects to a Third-party Proxy that is associated with the OCS and OSS/BSS. After an integration process, the resource joins the EFS. More information about integrating trusted and untrusted resources can be found in Section 4.4. The same interface T6 is also used by third-party organisations for deploying functions and applications on the EFS. Indeed, the target OCS/EFS shall not only provide third-party support for integrating external resources but also should be offer the capability of running third-party applications or functions on top its own resources. Ultimately, 5G-CORAL defines the interface F2 which is used for OCS-OCS communication. When a resource, either trusted or untrusted, is integrated into the EFS, the OCS may notify any peer OCS (if any) about the new available resource as to update their view of peer domains.

3.5.1 Federation and pricing insight

From the perspective of resource sharing, the role of each participating administrative domain in a federation can be categorised as follows:

1. Provider-only EFS acting as resource provider. It provides resources to other participated EFSs, but never consumes resources of any other EFS;
2. Consumer-only EFS acting as resource consumer. It consumes resource of other EFSs but does not provide resources to other participants;
3. Two-way EFS acting both insourcing and outsourcing roles in federation.

The EFS may explicitly announce its role to other participants, which helps reducing signalling traffic when the EFS is either a provider-only or a consumer-only EFS. Alternatively, the EFS may remain to be an implicit two-way participant to maintain flexibility. A rational two-way EFS should be willing to contribute and utilise the shared resources [44]. If a market-based dynamic pricing model is used for resource sharing, provider-only and consumer-only EFSs are essentially sellers and buyers in the market, respectively. Meanwhile, centralised broker node may keep the role of nodes and the status of resources. It depends on the designer's policy to delegate one federation role to broker to deal with. EFS operator can consume the resources from other administrative domains in the case of high input traffic to satisfy the SLA requirements and maintain the required QoS. On the other hand, due to economic incentive, EFS operators would be motivated to rent the idle (not used) resources to the other participating domains. More considerations on these business perspectives are under discussion in WP1. In addition, several possible charging plans are needed an envisioned for 5G-CORAL federated EFS. Static EFS federation enjoys the benefit of monthly or annually charge, which means resource consumers need not pay on a per-transaction basis. The fixed (annual or monthly) charge makes sense since the agreements are settled in advance, prior to establishing the network connections among the federated administrative domains. In dynamic

EFS federation, on the other hand, resource consumers may need to pay per usage because the federation relationship is dynamic and may not exist for long.

Depending on whether the unit price of resource changes over time, the pricing models can be categorised into fixed and dynamic. In case of fixed pricing, the unit price of each resource type in an EFS is fixed and predetermined. For static EFS federation, the price depends on contract signed by involved parties and also on the type of use. It is also possible that all involved parties agree a mutual waiver policy (resource users are exempted from paying the charge in the sake of sharing resource mutually). In case of dynamic EFS federation, a posted-price model can be used, where sellers or buyers announce a fixed unit price of the resource they are willing to sell out or purchase. Then parties decide whether or how much of the resource capacity they are willing to purchase or sell. Because neither buyers nor sellers have the obligation to maintain a federation relationship with each other, the posted prices affect (and could be a dominant factor in) federation formation. In fact, the federation relationship may be dynamically built based on trading. Applying price regulation mechanisms may keep the incentive of the administrative domains with low resource capacity to participate in the federation as two-way EFS.

Generally speaking, fixed pricing is not suitable to dynamic federation because the selling price cannot adapt to the dynamics of demand and supply situation. It can lead to low user welfare and load imbalance from the perspective of resource provider [45]. In case of dynamic pricing, the unit price of each resource is not fixed but negotiated in some way between the selling and the buying parties. Only after both parties agree on a price, the EFS provider can share the EFS resources with the EFS consumer. Dynamic pricing is more economically efficient than fixed pricing because resource price is set according to the forces of demand and supply [46]. If both resource consumer and provider already form a static EFS federation, either party can initiate a price negotiation using some negotiation protocol. The negotiation procedure can be done periodically (in a daily basis, for example) or in an on-demand basis. Because both parties are in a federation, the price may include some discount or amount of payment reduction for their long-term interests.

If the resource consumer and provider do not form a static EFS federation, then they can either negotiate with each other directly or participate in a resource exchange market or an auction. In resource exchange market, sellers, buyers or both can use a posted-price model to announce their anticipated unit prices. The posted-price scheme here differs from the same scheme used in the fixed-pricing model in that sellers or buyers here may dynamically change their posted prices to reflect their valuations on resource (which may depend on the supply and demand conditions). Dynamic Pricing Trade Market [49] is a type of dynamic pricing used in federated cloud. It is based on utilisation pattern of participating resources. The profit rate of shared resources is adjustable dynamically and it is proposed to be implemented in Internet Innovation Union (IIU) federated cloud. An on-line resource auction can also be used for dynamic pricing in federated EFS. Participants of an auction include bidders or buyers (i.e., resource consumers), sellers (e.g., EFS resource providers) and the auctioneer. One OCS can play the role of auctioneer that conducts the auction. An independent 3rd party may also play the role of a broker to facilitate price information exchange and/or winner determination in the resource auction. Table 3-7 summarises all possible charge plans in federated EFS environment, which 5G-CORAL can benefit.

TABLE 3-7: POSSIBLE CHARGING PLANS IN 5G-CORAL FEDERATED EFS

Federation Scheme	Static Pricing	Dynamic Pricing	Charging scheme
Static	• Depending on contract and type of use • Mutual waiver	• Posted-price scheme • Possible discount or certain amount of reduction	• Monthly or yearly subscription
Dynamic	• Charge based on usage • Price-dominant federation formation	• On-line market-based • Auction-based	• Pay per usage

Resource exchange market or auction is logically centralised but physically distributed. The reason is that EFS resources provided or to be consumed exhibit locality property as exemplified below:

- Some EFS resource is only accessible to "local" consumers. An example is wireless spectrum resource;
- EFS consumers may only have interest in the locally-accessible resource. An example is virtualised computation resource requested by low-latency applications like VR/AR.

In these cases, potential resource providers and consumers tend to cluster together, and it is nature for them themselves to locally exchange or share EFS resource. A global resource exchange or auction does not make sense in these scenarios.

Dynamics of Service Selection is kind of dynamic pricing, arising from the fundamental issue of cloud service selection while keeping the performance from cloud service provider's side [48]. EFS federation can benefit from this pricing model. Do et al. [49] proposed pricing model based on dynamic service selection in federated cloud. The price competition among the participating resource provider is based on how to sell the cloud services; in non-cooperative way to obtain the maximum revenue (i.e., provider's side) and receive the selected services with minimum latency and the best profit (user's side).

Apart from non-cooperative approach, the cooperative game pricing model can be applied in EFS federation which can motivate the EFS operators to contribute the resources. Regard to EFS federation, using cooperative game pricing model provides cost-effective resource supply that can make EFS operators keep their participation. Also, the more contribution in federation the more total utility in federation [49]. Hassan et. Al. [50] proposed dynamic pricing model for distributed resource allocation in federated cloud using cooperative and non-cooperative game. Cloud Asset Pricing Tree (CAPT) [50] is an elastic economic model which applies to participating federated cloud providers. It provides an optimal premium price of the Cloud federation options efficiently. CAPT aims to mitigate the risk of extra payment in the case of SLA violation. CAPT can be exploited for federated EFS pricing. EFS federation can be categorised into three models:

- **Trusted model:** it is similar to the client/server relationship where the client requests function/service to deploy in the target EFS and the server implements it. Meantime, to form this model, client authentication is needed. The server is responsible for management of the EFS resource and the client is responsible of the function/service lifecycle and/or actions.
- **Loan model:** it is corresponding to the lender/borrower relationship. The lender lends resource to the borrower. And, the borrower implements its function/services. A strong protection in terms of resource authorisation is demanded for establishing such a model.
- **Concession model:** it is similar to the supporting multi-tenancy model where, resource is totally managed and controlled by the client, with regard to a lease. Resource isolation scheme is needed for establishing concession model.

4 Resource discovery and integration

Discovery protocols allow devices and services to automatically become aware of the functionality and identity of other devices and services in the network without the need for human intervention. For example, a low power temperature sensing device may use a service discovery protocol to query other devices in the same area in order to determine which ones implement a heater switch service. This eliminates the need for an operator having to explicitly introduce this information to the sensing device. Several protocols have been proposed to enable service discovery between wireless/wired networked devices. However, since these protocols were not designed to be power efficient, they introduce significant overhead when adopted for constrained networks.

In fact, in the context of constrained networks, the following additional design requirements must be considered for resources discovery protocols:

- Low overhead per control message and reduced number of message exchanges to reduce energy consumption and save communication bandwidth.
- Low memory and processing requirements so that the discovery mechanism can run even in highly resource constrained nodes.
- Robust service provisioning to account for the high dynamics of the wireless communication channel and the unpredictable availability of battery powered devices.
- Interoperability with web applications and IP based backend networks so that these low power devices do not operate in isolated islands.

Given the above design requirements, new mechanisms should be developed to enable service and device discovery in constrained networks.

4.1 Discovery protocols – An overview

The main functionalities for a discovery protocol during its lifecycle are publication, registration into the directory (if available), discovery (in terms of browsing), and resolution. The goal of resource discovery protocols is to allow services and devices in the network to automatically become aware of each other without the need for human intervention. This section provides an overview of discovery protocols available in the literature. Figure 4-1 shows a taxonomy of the analysed discovery protocols while Table 4-1 provides a summary of each protocol along with its network requirements and scope. A more comprehensive description of the various protocols can be found in Appendix A.

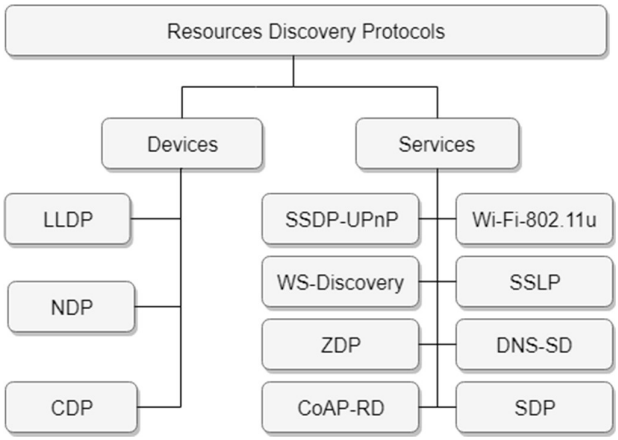

FIGURE 4-1: RESOURCES DISCOVERY PROTOCOLS TAXONOMY

TABLE 4-1: SUMMARY OF RESOURCE AND SERVICE DISCOVERY PROTOCOLS

Protocol	Description	Requirements	Scope
SSDP [A.1]	Text-based protocol for advertisement and discovery of network services and presence information that does not require any configuration a priori on the network nodes.	L2 connectivity IPv4/IPv6 ucast & mcast UDP HTTP	Local Area Network
UPnP [A.2]	Set of networking protocols for seamless discovery of networked devices and establishment of network services for data sharing, communications, and entertainment.	L2 connectivity IPv4/IPv6 ucast & mcast UDP & TCP HTTP & SSDP	Residential networks
IEEE 802.11u [A.3]	IEEE 802.11u-2011 is an amendment to the IEEE 802.11-2007 standard to add features that improve interworking with external networks including service advertisement, discovery and pre-association communication.	IEEE 802.11 connectivity	IEEE 802.11 networks
Wi-Fi Direct Discovery [A.3]	Wi-Fi Direct is a Wi-Fi standard enabling devices to easily connect with each other without requiring a wireless access point.	IEEE 802.11 connectivity IEEE 802.11u	IEEE 802.11 networks
Service Location Protocol [A.4]	Enables automatic service discovery in an IP-based network without prior configuration.	L2 connectivity	Personal Area Network
WS-Discovery [A.5]	A specification to dynamically discover services directly by clients in an ad hoc network or by using a discovery proxy as a broker to centralise the discovery.	IP multicast SOAP & UDP SOAP & HTTP	Ad-hoc network up to more extended ones (proxy needed)
LLDP [A.6]	LLDP is an industry-standard vendor-neutral method to allow networked devices to advertise capabilities, identity, and other information onto a LAN for device and topology discovery.	L2 connectivity	Local Area Network
DNS-based Service Discovery [A.7]	DNS Service Discovery is a way of using standard DNS programming interfaces, servers, and packet formats to browse the network for services.	L2 connectivity IPv4/IPv6 ucast & mcast UDP/TCP	Local Area Network
Neighbour Discovery Protocol [A.8]	Neighbour discovery is a protocol that allows different nodes on the same link to advertise their existence to their neighbours, and to learn about the existence of their neighbours.	L2 connectivity IPv6 ucast & mcast	Local Area Network
Cisco Discovery Protocol [A.9]	Cisco Discovery Protocol (CDP) is a proprietary Data Link Layer protocol developed by Cisco Systems. It is used to share information about other directly connected Cisco equipment, such as the operating system version and IP address.	L2 connectivity	Local Area Network
Simple Service Location Protocol [A.10]	Simple Service Location Protocol (SSLP) provides a framework for the discovery and selection of the services working on 6LoWPAN whilst being interoperable with IP networks.	L2 connectivity	Personal Area Network
ZigBee Device Profile	ZigBee device discovery provides the facility for devices to find node-wide (not application/ endpoint specific) information about other devices in	L2 connectivity RF4CE/ 6LoWPAN	Ad-hoc networks

[A.11]	a network, such as addresses, manufacturer ID, types of applications running, power source, and sleep behaviour.		
Service Discovery Protocol [A.12]	SDP is another wireless low power protocol, but mainly used in mobile ad-hoc environments such as between a mobile phone and headsets or car stereo systems. Bluetooth device discovery is performed by periodically broadcasting and scanning for inquiries.	L2CAP layer	Ad-hoc networks
Preboot Execution Environment [A.13]	PXE uses client-server protocols such as DHCP and TFTP to boot operating systems and install software retrieving from the network on client machines. PXE reduces the cost of supporting physical equipment.	L2 connectivity IPv4/IPv6 DHCP TFTP	Local Area Network
Zeroconf [A.14]	Set of protocols and techniques for automatic network configuration in a local area network. The three main tasks comprising zeroconf are address auto configuration, name-to-address translation and service discovery. For address auto configuration, zeroconf uses link-local addressing to replace DHCP.	L2 connectivity IPv4/IPv6 mDNS	Local Area Network
IoT COAP [A.15]	IoT COAP is a request-reply protocol enabling constrained IoT devices to communicate via the Internet, using UDP and a light-weight application layer.	L2 connectivity IPv4/IPv6 ucast & mcast UDP	Internet

Although the majority of these protocols is successfully employed to efficiently and reliably discover resources in a network, their scope is typically limited to Local Area Networks and residential networks. As an example, DNS-based service discovery is largely adopted to browse the network for services, yet it is mostly suitable for LANs, which makes it inappropriate for discovering resources located in heterogeneous networks, as with fog nodes due to their volatility and limited availability. Further room for discussion on the choice of the right discovery protocol will be left to Section 4.3.

4.2 Use case mapping in discovery and integration procedures

This section illustrates different procedures for discovering and integrating Edge and Fog resources into a target EFS depending on the specific use case. Multi-RAT aspects are also addressed to allow the discovery and integration of the resources across multiple access technologies. Furthermore, for each use case, the available resources are described, and a high-level message flow diagram highlighting the interactions between resources, EFS and OCS, is shown.

4.2.1 Cloud robotics use case

Cloud robotics is a field of robotics that leverages and integrates cloud computing into industrial and commercial robot applications as to enable robot systems to be endowed with powerful capabilities. Nonetheless, cloud facilities traditionally reside far away from the robots and nowadays cloud technologies present several limitations when applied to latency and jitter-sensitive cloud robotics applications. Indeed, while the cloud providers can enforce SLAs in their infrastructure, very little can be ensured in the network between the robots and the cloud. Given the pervasiveness of computing and networking resources in today's environments, a natural evolution of cloud robotics paradigm would be to also integrate the surrounding devices (e.g., computers, servers, access points, routers, base stations, etc.) into an integrated edge and fog platform that can be hence used to accomplish variegate and critical tasks for the robots, especially the ones with stringent latency requirements.

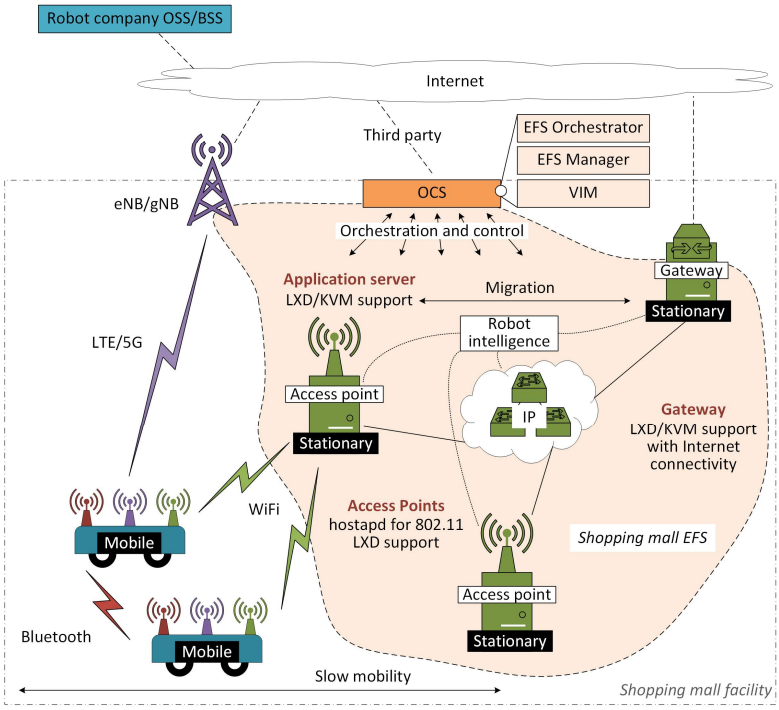

FIGURE 4-2: RESOURCES IN THE SHOPPING MALL FOR THE CLOUD ROBOTICS USE CASE

In this section, we analyse how the edge and fog resources envisioned in the 5G-Coral cloud robotics reference scenario [34] can be discovered and integrated into the EFS system for overcoming current cloud computing limitations. Figure 4-2 shows the reference figure of cloud robotics applied in the shopping mall where two scenarios are envisioned: in the first scenario the robots are in charge of keeping clean the floors in common areas of the shopping mall, thus providing a cleaning service; in the second scenario the robots provide synchronised delivery of goods within the shopping mall building to restock the supplies of the several shops. Table 4-2 reports the resources considered in the scenarios above also highlighting the corresponding ownership, connectivity technologies, and virtualisation support.

TABLE 4-2: RESOURCES INVOLVED IN THE CLOUD ROBOTICS USE CASE

ID	Type	Description	Scope	Owner	Conn.ity	Virtualisation
CR-1	Fixed	A server residing in a small/medium/large Edge DC	Host a virtual function providing Internet access	Shopping mall	Ethernet	VMs, Containers
CR-2	Fixed	A server residing in a small/medium/large Edge DC	Host the control robot logic	Shopping mall	Ethernet	VMs, Containers
CR-3	Fixed	Small computing devices distributed across the shopping mall equipped with 802.11 wireless cards	Host the virtual access point functions to enable the communication between the robots and the control logic	Shopping mall	Ethernet, 802.11	Containers
CR-4	Mobile	Robots equipped with battery, wheels, and motors capable of	The actual robots performing the cleaning and delivery actions	Robot company	802.11, Bluetooth, LTE/5G	Containers, Native

		roaming within the shopping mall	within the shopping mall			
CR-5	Fixed	Macro base station outside the shopping mall/Small cell in the shopping mall	LTE/5G connectivity is used by the robot for the initial connection with the robot company OSS/BSS	Operator/ Shopping mall	Ethernet, LTE/5G	--

In the following we describe the procedures for discovering and integrating the robots (CR-4) in the shopping mall EFS. It is important to highlight in this use case that the robots (CR-4) belong to the robot company and not to the shopping mall owner, who instead owns the EFS resources CR-1, CR-2, CR-3, and CR-5. Therefore, additional mechanisms are required during the discovery process as to ensure the proper integration of resources between the two involved players: the robot company and the shopping mall owner. As a result, this use case envisages the interaction of also the OSS/BSS layers in the discovery and integration process. Figure 4-3 shows the high-level message flow between the different components involved in the process. Please note that the Robot OSS/BSS may know in advance the location (e.g., URL) of the Shopping Mall OCS or alternatively it can discover it on-demand. This depends on the relationships and agreements between the two parties.

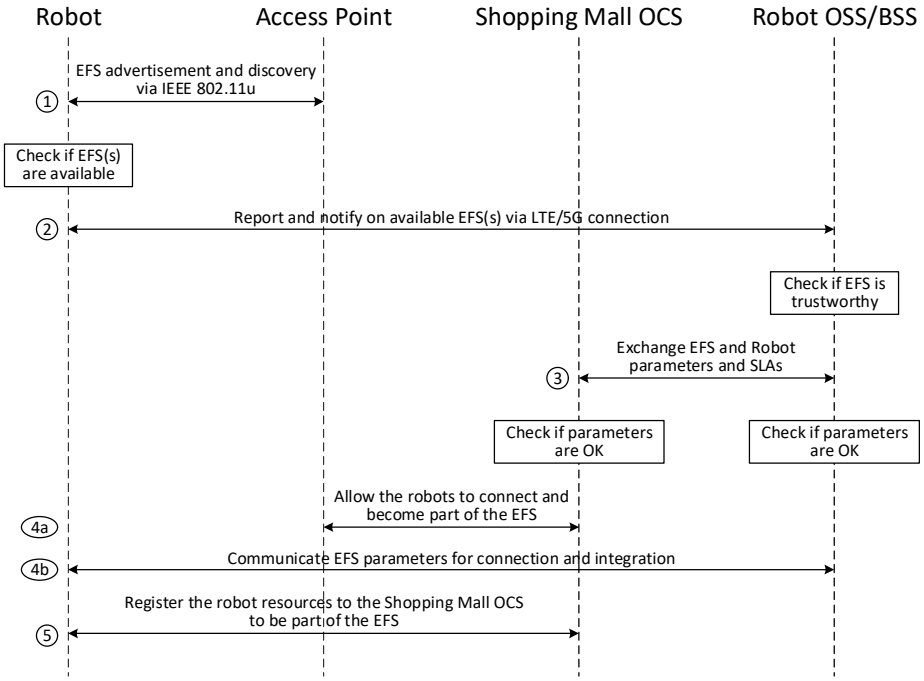

FIGURE 4-3: HIGH-LEVEL MESSAGE FLOW FOR ROBOTS INTEGRATION IN SHOPPING MALL EFS

4.2.2 Augmented Reality (AR) use case

The AR Navigation use case envisions a continuous indoor AR navigation experience for the clients at the shopping mall. The objective is to augment the user recorded video frames with a navigation arrow similar to the popular car navigation application. The user will see a guiding line grounded in the real-world image displayed on his screen so that it will remind a real object, a pointer, to the desired destination. Moreover, users will be able to see shop promotions on their screen

whenever they pass by the store. These special offers will enhance the shopping experience for the mall's clients.

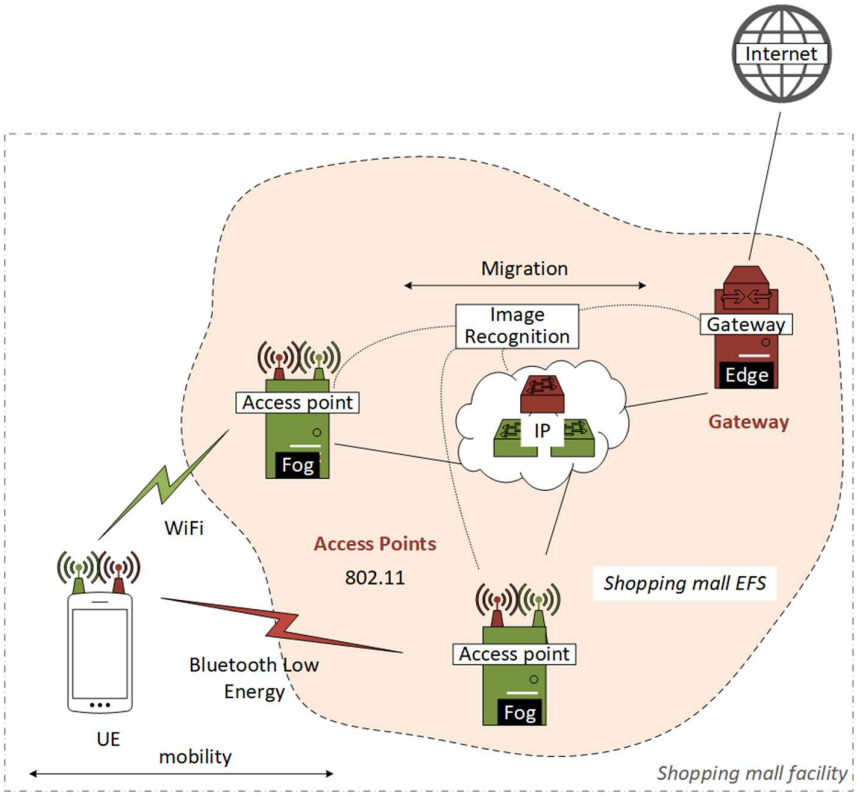

FIGURE 4-4: RESOURCES IN THE SHOPPING MALL FOR THE AR NAVIGATION USE CASE

The environment described in the AR Navigation use case drastically decreases the need for the video frame to travel from user's phone all the way to the remote data centre. Fog Computing Devices (CD) coupled together with shopping mall's Wi-Fi access points are deployed directly in the shopping mall, which places them very close to the end user. Networked Fog nodes can replace computing capability of the remote data centre. Figure 4-4: depicts the AR Navigation use case setup including involved resources. UE belonging to the shopping mall client constitutes a vessel to AR Navigation application (a non-EFS application). The application goal is to find a connection to the Image Recognition application (EFS application) deployed on the Fog CDs distributed around the shopping mall. Each Fog CD is coupled with a Wi-Fi access point controlled and iBeacon, both controlled by the OCS. While iBeacon is used to broadcast messages, which help to infer the location of the mobile phone, the Wi-Fi AP allows for basic connectivity of the UE with the rest of the network. All the resources included in the use case are summarised in Table 4-3.

TABLE 4-3: RESOURCES INVOLVED IN THE AUGMENTED REALITY NAVIGATION USE CASE

ID	Type	Description	Scope	Owner	Connectivity	Virt.lisation
AR-1	Fixed	A server residing in a Fog CD	Hosts the image recognition engine	Shopping mall	Ethernet, 802.11	VMs, Containers
AR-2	Fixed	Small computing devices distributed across the shopping mall equipped with 802.11 wireless cards	Hosts the virtual access point functions to enable the communication between the UE and the EFS entities	Shopping mall	Ethernet, 802.11	Containers

AR-3	Fixed	iBeacon	Broadcasts beacon signal	Shopping mall	BLE	-
AR-4	Mobile	UE	UE receives iBeacon signal and communicates with the Wi-Fi AP. It hosts AR Navigation application which displays AR content	3rd party	Wi-Fi, BLE	-

In AR Navigation use case the need for discovery is limited. Besides the UE (AR-4), all other resources are integrated in the shopping mall substrate belonging to the shopping mall owner. Therefore, the environment is perceived to be trusted. Discovery in such environment can be performed in an owner specific manner and is not explored here any further. What follows, no dynamic discovery is assumed for AR-1, AR-2 and AR-3 type of resources. Since AR-4 is not an EFS resource and does not require interaction with the OCS discovery in the AR Navigation use case, we will then refer only to the cases in which the UE accesses the Wi-Fi AP (AR-2) or receives signal from iBeacon (AR-3). The general procedure for Wi-Fi AP discovery is related to broadcasting a probe request and receiving probe responses form the surrounding APs. On the other hand, the UE will scan the area for the beacon signal from the iBeacon devices (AR-3). An information related to beacon ID (UUID) and signal strength of the discovered iBeacons is then extracted for further use. Both mechanisms are depicted in Figure 4-5. After the UE gains access to the network, integration of AR application with EFS system can take place. This mechanism, even though shown in Figure 4-5:, is out of scope of the present document. For more detail please refer to [33].

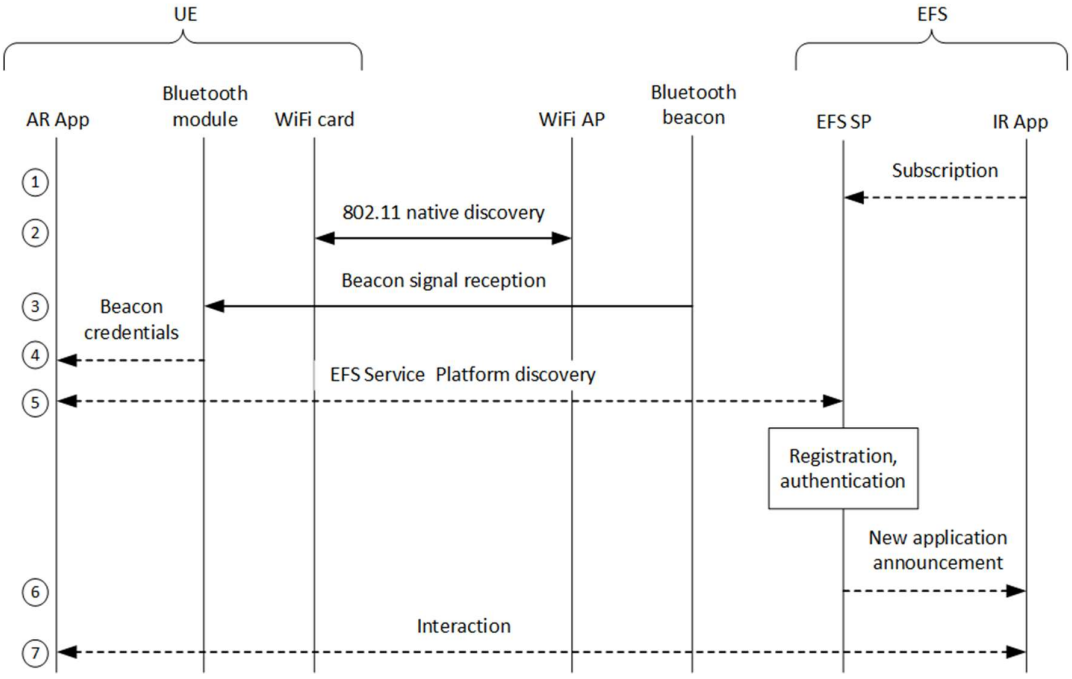

FIGURE 4-5: HIGH LEVEL MESSAGING FLOW FOR THE ACCESS POINT AND BEACON SIGNAL DISCOVERY (SOLID LINE) AND CONSECUTIVE AR APP INTEGRATION INTO EFS SYSTEM (DOTTED LINE)

4.2.3 Vehicular use case

The connected-car (or vehicular) scenario consists of two different sub-scenarios, i.e., vehicular communications and infotainment, which aim to show key economic and societal benefits of next-generation connected cars. Figure 4-6 shows the network topology and points out the connections

between multiple entities, i.e., cars, fog CDs, edge DCs and the backend cloud, characterised by a different level of end-to-end latency. . We assume that cars are equipped with On-Board Units (OBU) characterised by a certain computation power and a range of radio access interfaces, such as Wi-Fi, 3GPP LTE, GPS, etc., which enable car-to-car and car-to-infrastructure communications, whereas Road-Side Units (RSU) are installed on lampposts or traffic lights and provide OBUs with point of access to the underlying infrastructure.

The first sub-scenario includes vehicles periodically broadcasting information, such as position, speed, trajectory, diagnostics, etc., as well as traffic conditions and road hazards through edge devices distributed along the road. This information can be processed in order to generate warnings and help drivers make effective decisions. These services require tight collaboration between vehicles and edge devices, low latency and high computational capacity. Furthermore, collected information can be used to reserve a parking slot or pushing car software updates.

The second sub-scenario describes heavily traffic congested situations, where video streaming and web browsing services are massively demanded. In this context, MEC can contribute to addressing the increasing bandwidth demand and signalisation overhead, along with fog CDs.

The goal of the discovery process in the vehicular scenario is to discover and aggregate new EFS resources to the EFS domain, thus allowing the corresponding OCS to conveniently manage and allocate it. The discovery procedure shall be carried out in the plug-and-play fashion regardless of the initial configuration of the computing resources. A detailed description of the resources discovered by the OCS is provided in Section 3.4. Resources available in the vehicular scenario are summarised in Table 4-4.

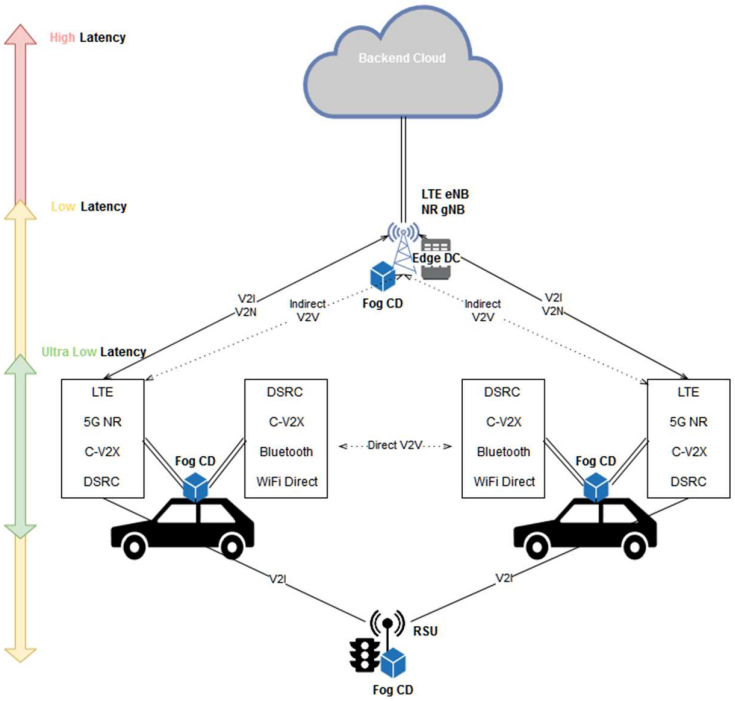

FIGURE 4-6: TOPOLOGY OF THE VEHICULAR SCENARIO

TABLE 4-4: RESOURCES AVAILABLE IN THE VEHICULAR SCENARIO

ID	Type	Description	Scope	Owner	Conn.ity	Virtualisation
V2X-1	Fixed	A server residing in a small/	Host functions, service platforms	Network operator/ Local municipality	Ethernet	VMs, Containers

		medium/large Edge DC	and applications			
V2X-2	Mobile	Small computing devices (OBU) installed in the cars with Multi-RAT capabilities (LTE/5G-NR/ DSRC)	Host virtual access point functions to collect and process real-time info	Car maker	LTE, 5G-NR, DSRC	Containers
V2X-3	Fixed	LTE/5G-NR small cells and DSRC RSUs	LTE/5G-NR/DSRC connectivity	Network operator	Ethernet, LTE/5G-NR	--

EFS discovery and integration of fog nodes (deployed at OBUs) located in connected vehicles can be performed in two separate ways:

1. **Direct discovery (V2I)**
   When the OBU is within a gNB/eNB/RSU radio coverage, V2I connectivity is employed. In this respect, the fog nodes can use an upper IP layer discovery protocol to join the EFS, once a PDN (Packet Data Network) connection is established between the terminal and the cellular network. Figure 4-7 illustrates how a car connected to an AP interacts with EFS and OCS in order to take part in the EFS.

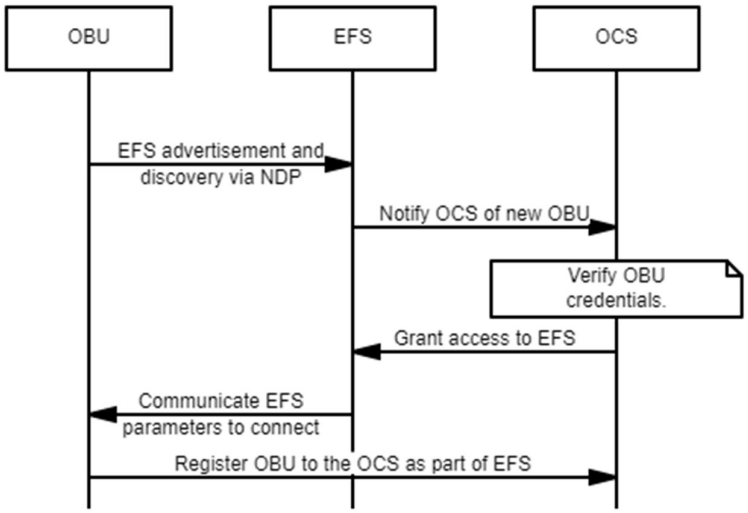

FIGURE 4-7: HIGH-LEVEL MESSAGE FLOW FOR DIRECT DISCOVERY SCENARIO

2. **Relay-aided discovery (V2V)**
   An OBU trusted neighbour can act as mobile relay to help an out-of-coverage fog node attach to the EFS. DSRC V2V and/or 3GPP V2X technology can be employed to establish D2D connectivity between two vehicles. More specifically, 3GPP ProSe (Proximity Service) can be adopted to allow vehicles to detect each other and establish direct connectivity. Furthermore, two different modes are supported:
   1) *network-independent direct communication*, where network assistance is not needed for connection authorisation, and communication is performed by only using local information and functionality;

2) *network authorised direct communication*, where network assistance is always required for connection authorisation, and it is typically performed when the terminals are served by the same access network.

A high-level message flow diagram describing the relay-aided discovery triggered by an out-of-coverage car is shown in Figure 4-8. Specifically, the diagram refers to the network authorised direct communication mode, where the car requesting access must first obtain access to the 3GPP network and then complete the EFS discovery procedure.

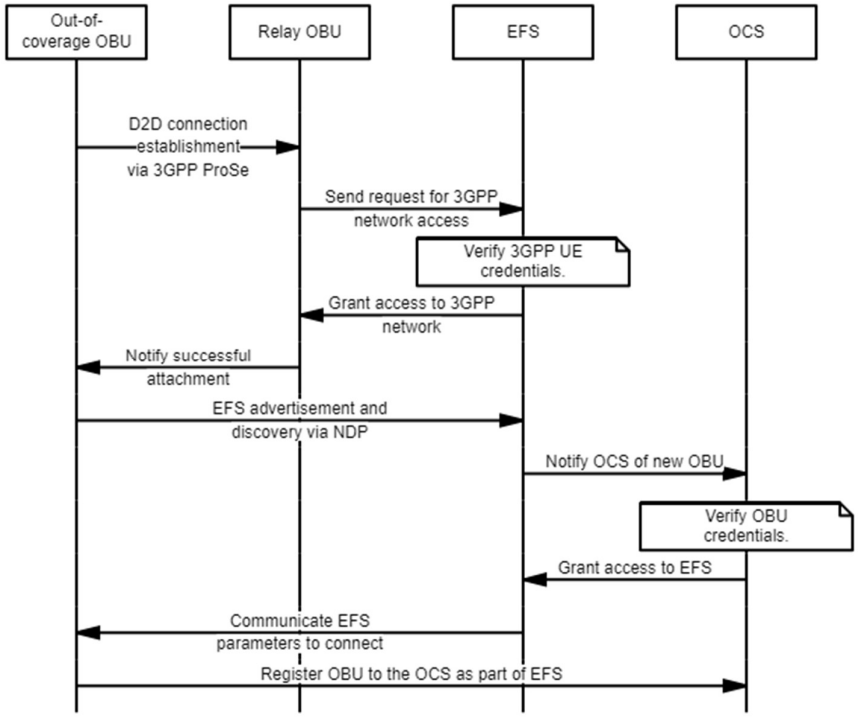

FIGURE 4-8: HIGH-LEVEL MESSAGE FLOW FOR RELAY-AIDED DISCOVERY SCENARIO

Further details on the 3GPP resource discovery mechanisms are presented in Section 4.3.3.

4.2.4 High-speed train use case

A high-Speed Train connects major cities in Taiwan with a speed up to 300 km per hour along more than 400 km railroad. Passengers on board are seeking for service continuity during the long trip or when they stop at given station. In 5G-CORAL project, EFS units deployed on board and on land of high speed train and train station respectively will provide breakout and mobility functions. In Particular, on board unit has EPC deployed where vMME will allow to mitigate the burden of passengers' mobility signalling out from the backhaul. Consequently, EFS will provide a seamless connection in a very challenging signalling environment resulted from frequent handovers. Moreover, OCS can migrate content and UEs' context information form on-board EFS unit to on land EFS in advance when passengers are approaching to the train station.

In the high-speed train scenario, the objective of the discovery process is to aggregate EFS resources based on the train location. This will allow OCS to manage and allocate them accordingly. The discovery procedure shall be carried out in a plug-and-play fashion regardless of the initial configuration of the computing resource and it depend on preconfigure EFS resources. Resources available in the high-speed train are summarised in Table 4-4.

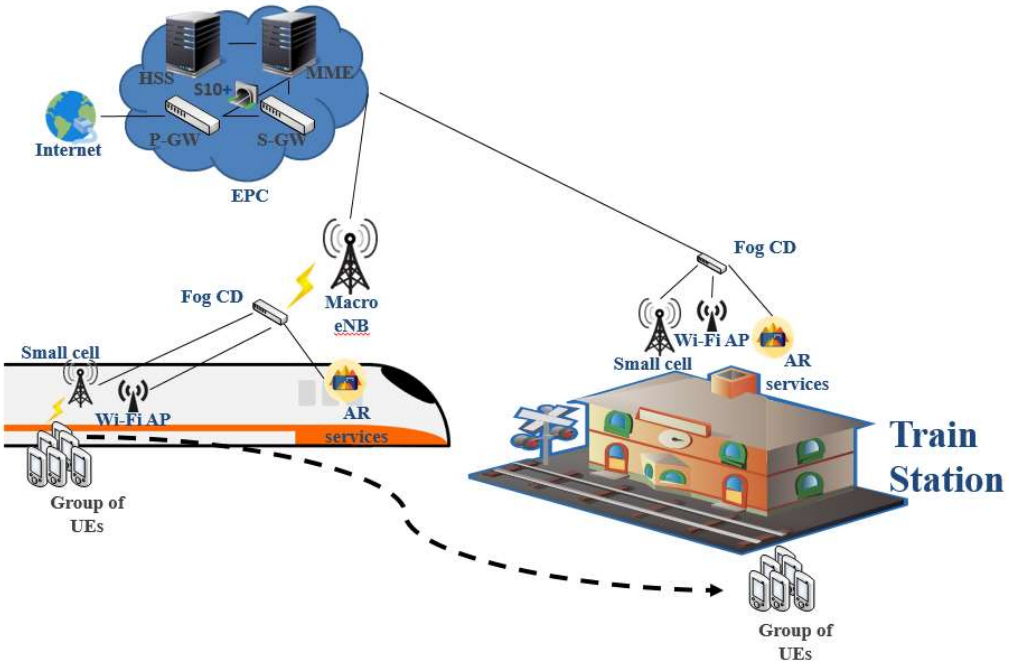

FIGURE 4-9: TOPOLOGY OF HIGH-SPEED TRAIN SCENARIO

TABLE 4-4: RESOURCES AVAILABLE IN THE HIGH SPEED TRAIN ENVIRONMENT

ID	Type	Description	Scope	Owner	Connectivity	Virtualisation
HST-1	Fixed	A server residing in a small/ medium/large Edge DC	Host functions, service platforms and applications	Network operator/ Local municipality	Ethernet	VMs, Containers
HST-2	Fixed for on-land and Mobile for on-board units	Small computing devices installed on-board in train or on land of train station has access to Multi-RAT network (LTE/802.11)	Host mobility management entity (MME) functions and/or break out traffic	Operator/ High-Speed train company	LTE, 802.11, Ethernet	VMs, Containers
HST-3	Mobile	Small cells installed on the train	LTE/5G connectivity	Operator/ High-Speed train company	Ethernet, LTE/5G	--
HST-4	Mobile	Wi-Fi Access point installed on the train	802.11 connectivity	Operator/ High-Speed train company	Ethernet, 802.11	
HST-5	Mobile	UEs on-board	UE communicate with the 802.11 AP and small cell. It hosts different application such	3rd party	802.11, LTE,	-

			as AR and video streaming			

The EFS discovery and integration of fog nodes located on the train can be performed in two separate ways:

- Direct discovery- neighbour's resources and element can be discovered through medium access such as Wi-Fi, Ethernet and LTE. In particular, IP layer discovery protocol for joining the EFS;
- Service discovery – Each device should be able to exchange services/information that is related to its capabilities or state. In this process, the OCS can discover further information in the exchange process. This can be achieved in many application level such as DNS.

In high-speed train use case, discovery is needed only for HST-5 (UEs of passengers) while other resources are integrated on board of train or on the land of the train station. We can say, no discovery is required for HST1, HST2, HST3 and HST4. Since these resources for high-speed train is deterministic. Only resources which are subject of discovery are the end user devices which consider as non-EFS resources (HST-5) in trusted domain. In addition, HST-5 does not interact with OCS while different applications are used. Consequently, HST discovery refers to interaction with HST3 (small cell) and HST4 (Wi-Fi) as illustrated in Figure 4-10 with minimum interaction with OCS. Needless to say, HST5 discovery mechanism as cell search or 802.11 native searches is well defined and in the act for commercial units. After HST5 gains access to the network integration of the certain application with EFS system can take place. This aforementioned mechanism is out of the scope of the present document. For more detail please refer to [33].

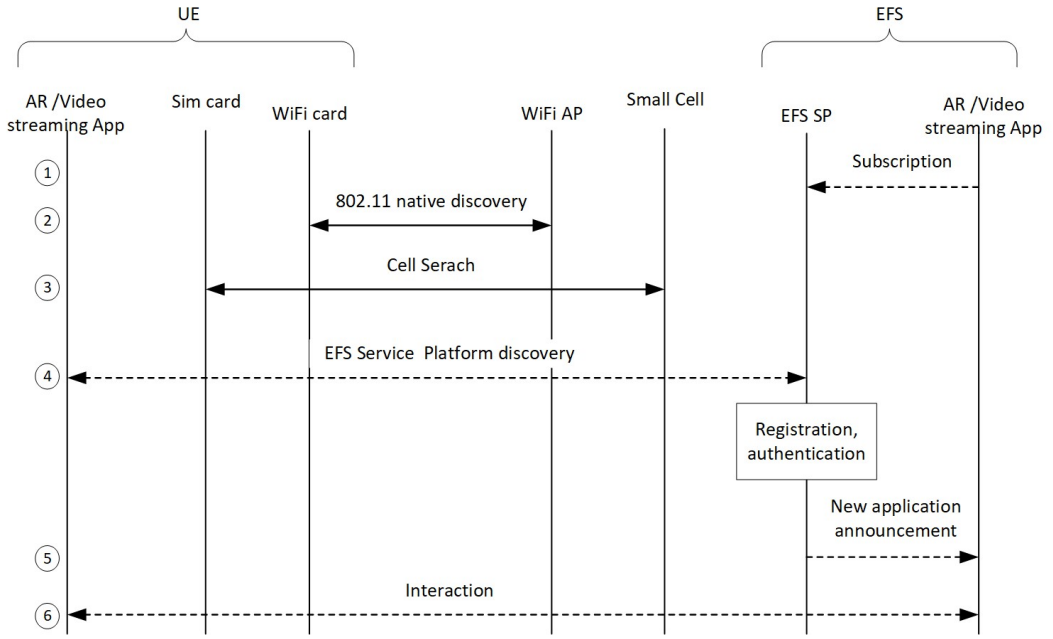

FIGURE 4-10: HIGH LEVEL MESSAGING FLOW FOR ACCESS POINT AND SMALL CELL DISCOVERY (SOLID LINE) AND CONSECUTIVE APPS INTEGRATION INTO EFS SYSTEM (DOTTED LINE)

4.2.5 IoT gateway use case

The number of Internet of Things devices is increasing continuously every year. These devices are usually, from a capability point of view, constrained devices with low processing or storage power, for instance. All these devices form a heterogeneous environment where many technologies are involved. Regarding the Radio Access Technology, one can find implementations of different ones in IoT devices, for instance, IEEE 802.15.4, Bluetooth Low Energy (BLE), LoRa, NB-IoT, ZigBee… Since the number of technologies increases, there is a need for deploy different access points that

support these communication technologies at the same time, thus optimizing the access to the network. Often only one technology cannot fulfil the requirements of complex scenarios like smart manufacturing, smart building, etc. Currently, parallel networks have to be deployed in this case.

This use case is expected to deploy IoT gateways in a shopping mall that support more than one Radio Access Technology. Hence, the above presented issue is reduced just implementing the proper virtual functions in the EFS. Heterogeneous devices are able to connect to the Edge via the IoT Gateway due to the centralisation of IoT functions from lower layers to higher layers into an edge computing platform, to create a convergent IoT system for multiples technologies as described previously. Figure 4-11 shows the expected resources that populate the IoT Gateway use case in the shopping mall scenario where IoT Access Points have been deployed.

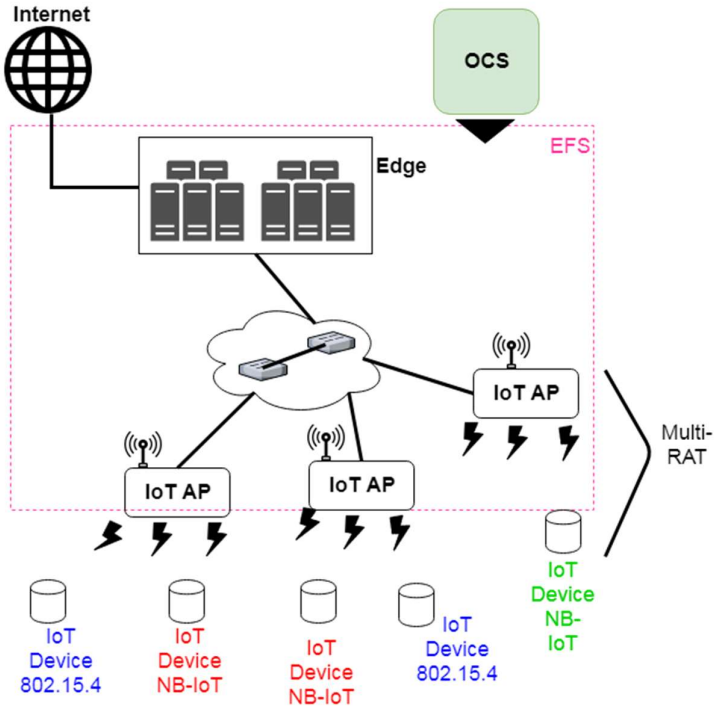

FIGURE 4-11: RESOURCE IDENTIFICATION IN IOT GW UC

In this section, the discovery of resources involved in IoT Gateway Use Case is analysed. Table 4-5 reports the resources identified in the use case.

TABLE 4-5: RESOURCE IDENTIFICATION FOR IOT GATEWAY

ID	Type	Description	Scope	Owner	Connectivity	Virt.lisation
IoT-1	Fixed	A server residing in a small/medium/large Edge DC	Host virtual functions providing Internet access, computation and storage.	Shopping Mall	Ethernet	VMs, Containers
IoT-2	Fixed	IoT Access Point	Provides access functionality for IoT devices. Form by a hardware device (antenna), and virtualised functions that process the raw data received.	Third party	Ethernet, Multi RAT	Containers

IoT-3	Mobile	IoT device	Devices performing some actions, used by users or other machines, and connected to the EFS.	Third party	IoT Wireless connection, e.g. BLE, 802.15.4, NB-IoT	Containers

The first resource whose discovery is needed to be performed is the IoT Access Point, which will afterwards allow the IoT devices to use the network. The IoT Access Point is expected to be connected, through an Ethernet network, to the EFS physical infrastructure. Since the owner of an access point could be the shopping mall one or a different one, different considerations arise at this point. The case of different ownership of access points involves a more complex discovery approach where probably an Operation Support System/Business Support System is part of the process managing integration agreements with EFS/OCS. Thus, the following figure shows a high-level message flow to discover and integrate the access points with EFS and OCS, without considering an OSS/BSS.

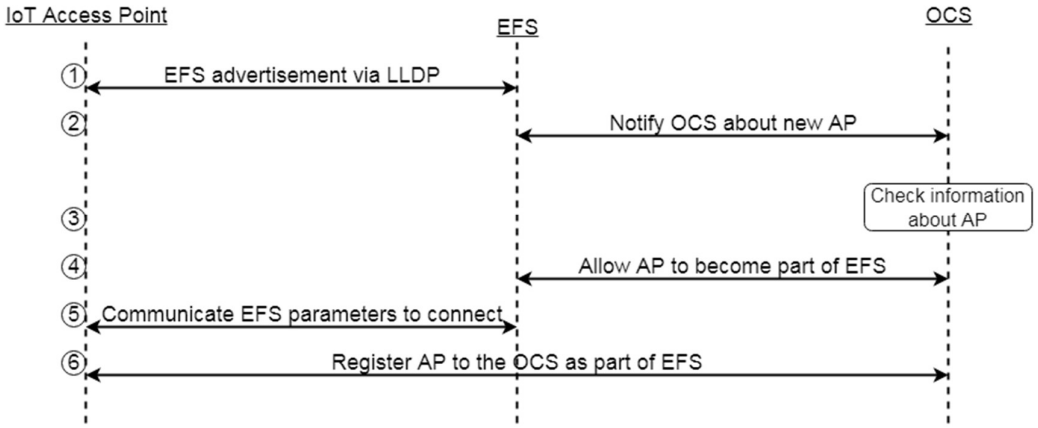

FIGURE 4-12: HIGH-LEVEL MESSAGE FLOW AP

Since the access points are connected to the EFS, it should be the first interaction. Then, the OCS has to start managing the access points, and finally register them.

4.3 Discovery of EFS

In an ever-changing environment, computing, networking and storage resources need to be added to the EFS system on the run. However, before an integration process can take place a discovery procedure needs to commence. In the dynamic environment, such as shopping mall, it is assumed that the new resources take an active role in the discovery process. Indeed, it is assumed a device (e.g., UE or robot) arriving to the local network will intend to discover available EFSs. Moreover, since the device may belong to a third party, it can autonomously choose which EFS it wishes to join. The decision can be motivated by the access technology used, services provided but also economic reasons.

One method for the device to discover available EFSs is reuse of network access protocols. Access protocols offer extended information about the network which can be acquired before a device engage into procedures preceding entering the network such as authentication. This provides tangible benefits to the device. With network information it can not only decide which network to access but also how to do it. For example, a device can decide which credentials to use for the authentication procedure. This mechanism can be reused and augmented to provide EFS and/or OCS information to the device. Based on that information the device can get to know which access network supports EFS of interest to later decide which credentials to use, to enter it. This discovery approach can bring considerable time savings. Thanks to acquiring EFS/OCS information in the

initial stage of deployment would reduce the discovery delay which otherwise would include accessing the network and using application level discovery protocols on top. Such low latency discovery is critical to latency sensitive applications. At the same time pre-authentication network assessment and EFS/OCS discovery leads to significant complexity reduction as the device not only can pick one of the available networks but also does not need to use application layer protocols to scan network looking for system elements. This feature is especially important for volatile resources.

Use of network access protocols as a discovery tool is dependent on the technology itself since it requires augmentation of each protocol separately with the information relevant to the EFS discovery process. The approach to augment existing access protocols together with the description of the relevant information exchanged with the device are listed and described in more detail later in the following sections.

4.3.1 Connecting via IEEE 802.11

Discovering the EFS via Wi-Fi interfaces can be done via IEEE 802.11u protocol which was ratified by the IEEE in February 2011. IEEE 802.11aq is a forthcoming amendment of 802.11u which extends some of the discovery functionalities. One of the motivations for this standard is to allow Wi-Fi client devices to learn more about a network before deciding to join it. IEEE 802.11u/aq allow indeed mobile devices to gain additional insight on the network capabilities to perform a better network selection, automated roaming and offload, secure authentication, and QoS integration with operator networks carrying user traffic. Those devices can use active (probes) or passive (beacons) scanning techniques to discover surrounding Access Points (APs), learn about the network, determine which network is best, and make a connection. Traditionally, this process depends on device recognition of the network name: the SSID. 802.11u/aq do not fundamentally alter the basic discovery process; however, they do enable the discovery of additional information during the scanning process and it allows the client to query the AP for additional information.

IEEE 802.11u defines the Generic Advertisement Service (GAS) framework with the objective of providing transport for advertisement services (see Appendix A.3 for more information). The main reason of using GAS is that prior to association, mobile devices have not obtained an IP address yielding to the impossibility of using IP-based discovery protocols (see 4.1 for a summary). In 802.11 networks, a device and AP may be in one of several different states of connectivity. The network discovery and selection process occur when the AP and device are in an un-authenticated and un-associated state. The 802.11 protocol has a special frame type that can be used in this first stage (un-authenticated, un-associated) to invoke a specific action by the recipient. This is called a Public Action frame. 802.11u introduces new Public Action frame subtypes for GAS requests and responses, enabling the client to prompt the AP into action before an association is formed. This is critical for advanced network discovery capabilities and other future advertisement services. At that end, the device sends a GAS Request, providing a list of information that it wants to receive from the AP. The AP either satisfies the query out of its own configured datastore or relays the client's query to backend advertisement servers. Please note that the backend encapsulation protocols are not specified by 802.11u. The AP replies to the device with a GAS Response, which includes the information requested by the client.

Since GAS provides a transport for advertisement services, many ways are available for encoding discovery information (e.g., UPnP, SDP, etc.). However, 802.11u mandates the support of the Access Network Query Protocol (ANQP) which defines a set of elements for uniquely identifying the information (see Table A-5 in Appendix A) [27]. Specifically, these elements are used by the devices to discover information that is not sent in beacons. In order for the devices to discover the EFS, additional fields are hence required as reported in Table 4-6. Such fields will be then used for enabling resource integration as described in Section 4.4.

TABLE 4-6: EFS-SPECIFIC ANQP INFORMATION ID DEFINITIONS

Info ID	Info name	Description
283	EFS URI	Provides a list of one or more URI for uniquely identifying the EFS available on the network

284	OCS Entry Point URL	Provides a URL for the Orchestration and Control System which can be used as entry point for integrating the resource into the existing EFS
285	EFS Service Information Request	Contains a generic request for EFS Service Information
286	EFS Service Information Response	Contains the detailed EFS Service Information in response to a EFS Service Information Request

4.3.2 Connecting via Ethernet

As described in Section 4.2.5, IoT Gateway Use Case requires wired Access point devices. Thus, it is needed a resource discovery protocol supporting IEEE 802.3 networks. As described in Appendix A, LLDP and CDP operate at low layers of the protocol stack (e.g. layer 2 bridges) and allow discovery of network resources. Since CDP is a proprietary protocol, this section will focus on LLDP to address the desired discovery. See Appendix A for more information.

The access point wired device will attempt to connect to the EFS within the discovery process. It may execute some integration service client to manage the subscription to the EFS Integration Service and send some context data, to be defined later in this section. This client could process some data when receiving a list of VIMs to select the best option. The wired device and VIM have to be connected at L2/L3 layer.

In Appendix A, TLV are explain. This field in LLDP frames allow to this protocol to extract the objects from the local MIB and be transported by it (TLV). There are some fixed TLVs, but it is possible to create custom TLVs to exchange information. This information exchanged allow to reach to different resources types in EFS and it allow to connect to OCS to manage those resources.

With respect to the organisationally specific TLV, the information to be transmitted has to be defined, such that it is understood by the EFS and OCS and therefore perform a successful integration. There should be a homogeneous information basis. Some of them are proposed in the table below.

TABLE 4-7: EFS-SPECIFIC LLDPDU INFORMATION TLV-TYPE DEFINITIONS

TLV-Type	Info name	Description
9	EFS URI	Provides a list of one or more URI for uniquely identifying the EFS available on the network
10	OCS Entry Point URL	Provides a URL for the Orchestration and Control System which can be used as entry point for integrating the resource into the existing EFS
11	EFS Service Information Request	Contains a generic request for EFS Service Information
12	EFS Service Information Response	Contains the detailed EFS Service Information in response to a EFS Service Information Request

From an implementation point of view, the usage of SDN networking could help with the discovery since not only LLDP is executed but Broadcast Domain Discovery Protocol (BDDP). The controller floods with BDDP packets and are broadcasted by non SDN network resources. With this, the discovery of direct and indirect links is achieved. Hosts are discovered by the controller by protocols like ARP or NDP. When the first packet generated by a host arrives at the switch, it does not have a rule installed to treat the package and is forwarded to the SDN controller, which discovers the host. Switches will also send a message to the controller when starts a connection with it, knowing a predefined IP and port. Please note that switches, links and hosts can be generalised as resources like devices, servers, storage, etc.

4.3.3 Connecting via 3GPP

In this subsection, we describe standard procedures to enable non-3GPP access in LTE systems as well as solutions under study for next-generation 5G systems. A state-of-the-art review on methods to enable trusted and untrusted non-3GPP access in 4G and 5G networks is also provided in

Appendix C. Finally, we present a mapping between trusted and untrusted non-3GPP access interfaces and the 5G-Coral EFS interfaces. Specifically, we consider a scenario where a connected car is willing to provide its computational and storage resources, though its non-3GPP RAT is considered untrusted by the EFS.

4.3.3.1 3GPP Access Network Discovery and Selection Function (ANDSF)

To assist UEs in the discovery/selection of 3GPP and non-3GPP access networks (Wi-Fi, LTE, WiMAX, etc. in their proximity, 3GPP introduced the Access Network Discovery and Selection Function (ANDSF), an entity within the EPC network which enforces operators' mobility policies for connecting to such networks. To this end, a collection of rules called Inter-System Mobility Policies (ISMP) are made available for the UEs to choose the most preferable RAT.

ANDSF was first standardised in 3GPP LTE Rel. 8 to facilitate the UE discovery of new non-3GPP access networks. Later on, with 3GPP LTE Rel. 10, the ANSDF framework was extended to consider simultaneous network connections to multiple RATs, introducing the Inter System Routing Policies (ISRP) module, which enables operator's policies based on the UE traffic. More specifically, ISRP can handle policies based on:

- Destination IP address the UE sends data to,
- Destination port number the UE connects to,
- The PDN identifier,
- A combination of the previous three elements.

Further developments to ANSDF were introduced in 3GPP LTE Rel. 11 in terms of better identifying the traffic to which a given ISRP applies, e.g., by locating a traffic portion with specific characteristics, yet associated with a port number typically belonging to different services (e.g., port 80 for HTTP). As a result, new metrics to identify the traffic were employed, ranging from IP flow throughput and file size, to the application name or identifier.

Along with managing ISMP and ISRP, ANDSF employs Access Network Discovery Information (ANDI) which comprises a list of access networks available within UE's surrounding, received as per UE's request, including its location and capabilities, e.g., supported RATs, radio access network identifiers (WLAN SSIDs), and technology-related specifications, such as number of available carriers.

Figure 4-13 shows the ANDSF architecture, where a UE can be registered to both the Visited-PLMN (VPLMN) and the Home-PLMN (HPLMN) and interact with the two networks. Such architecture is client-server, with the IP-based interface S14 between the UE and H-ANDSF/V-ANDSF distributing the network selection information and policies.

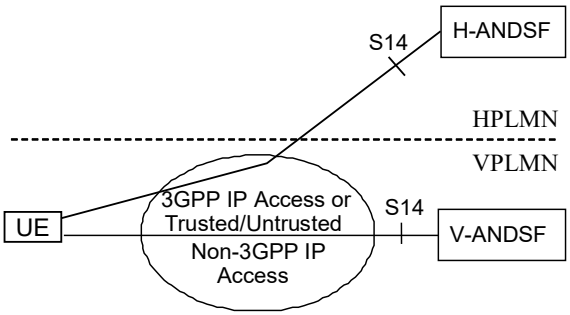

FIGURE 4-13: ARCHITECTURE FOR THE ANSDF PROTOCOL

Moreover, ANSDF features two communication models, namely, *pull model* and *push model*. In the former, ANSDF discovery is performed by the UE, which contacts the ANSDF in order to request policy information. Upon reception of this, the UE processes the response and enforces the policy.

By contrast, in the push model the UE receives the policy typically through the SMS protocol. More specifically, communication between the ANSDF client and server is managed through the **Open Mobile Alliance Device Management** (OMA-DM) protocol over the S14 interface. Based on a lighter version of XML called SyncML, OMA-DM consists of three phases:

1. *Alert Phase*, optional and used only in case of servers triggering the management sessions;
2. *Setup Phase*, handling authentication, and device information exchange. This phase is initiated by the UE through a setup request, including device information, client credentials and session origin, followed by a server response;
3. *Data (Management) Phase*, where messages are encapsulated in HTTP POST requests and contain status information, while the server can reply with HTTP ACK messages including commands or indicating that there are no more operations.

Moreover, setup and data phase is carried over a TLS (Transport Layer Security) session. Finally, device configuration data presents a hierarchical structure denoted as *device management tree*, where sub-trees are called device management nodes, and a single leaf represents a manageable object (MO). OMA-DM also provides a set of server commands to manipulate the Mos through SyncML messages, i.e., ADD, GET, REPLACE, DELETE, COPY, EXEC.

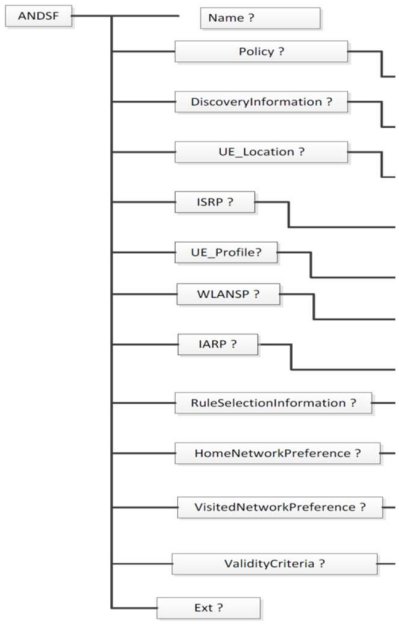

FIGURE 4-14: ANSDF DEVICE MANAGEMENT TREE

Figure 4-14 shows the device management tree with possible device management nodes, while a detailed representation of the *DiscoveryInformation* subtree is shown in Figure 4-15. We also denoted in red the proposed additional information required to advertise and discover EFS information.

More specifically, EFS URI presents a list of one or more URI to uniquely identify the EFS available in the network, while OCS entry point URL provides a URL for the OCS which can be used as entry point to integrate the resource into the existing EFS. It is also worth pointing out that ANDSF-based discovery of the EFS can be currently supported by 4G networks as described above, while ANDSF in 5G networks is still under study within 3GPP standardisation activity [60], thus support to EFS discovery through ANDSF is not yet available.

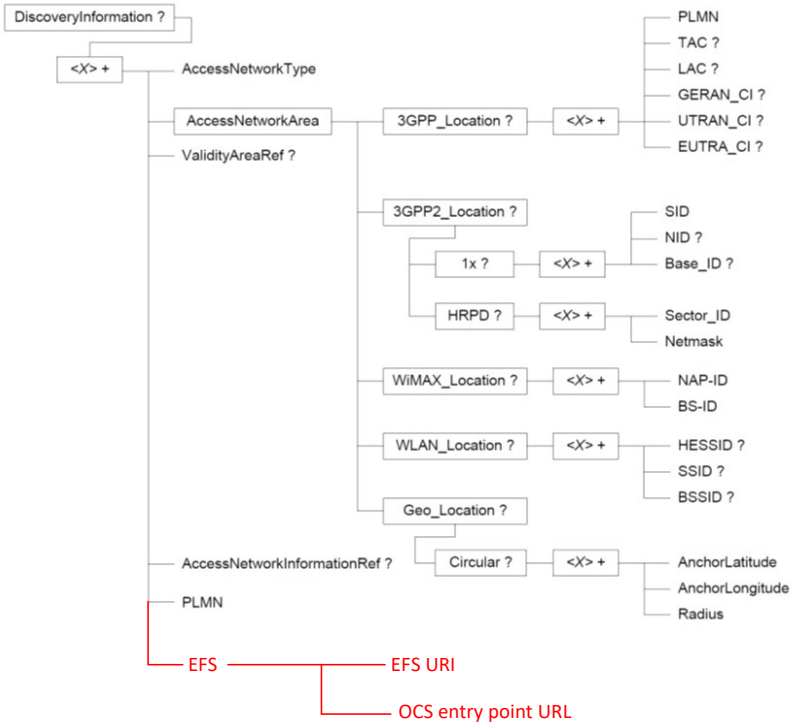

FIGURE 4-15: DISCOVERYINFORMATION SUBTREE WITH ADDITIONAL INFORMATION FOR EFS DISCOVERY

4.3.3.2 Mapping trusted and untrusted access onto 5G-Coral architecture

In Figure 4-16, car A is equipped with a 3GPP LTE RAT and associates with the 5G-Coral EFS via a 3GPP trusted domain, whereas car B wishes to connect to the 5G-Coral EFS through a non-3GPP untrusted RAT, e.g., DSRC. Moreover, we assume that car A and car B can directly communicate with each other via D2D connectivity.

To discover and integrate fog node resources available in car B, D2D connectivity first needs to be established. To this end, the 3GPP ProSe (Proximity Service) located inside the EFS Service Platform can be employed to facilitate the D2D discovery and synchronisation between the two cars. ProSe supports network-independent direct communication, requiring no network assistance, or network authorised direct communication, typically adopted when UEs use the same access network. In this example, we assume that car B does not have a 3GPP RAT, thus network assistance is not envisaged.

The next step consists of car B attaching to the NextGen core network located inside the EFS. As shown in Figure 4-16, the NWu interface responsible for enabling the attach procedure via the untrusted non-3GPP access is mapped onto the 5G-Coral T8 interface, which links the EFS service platform to Non-EFS Applications/Functions. Furthermore, N3IWF needs to be connected to the NextGen Core network through NG2 interface (to CP functions) and NG3 interface (to UP functions), which are both mapped onto the 5G-Coral E2 interface. At the end of the attach procedure, car B is able to communicate with N3IWF via an IPSec connection and to exchange NAS signalisation with the NextGen core network.

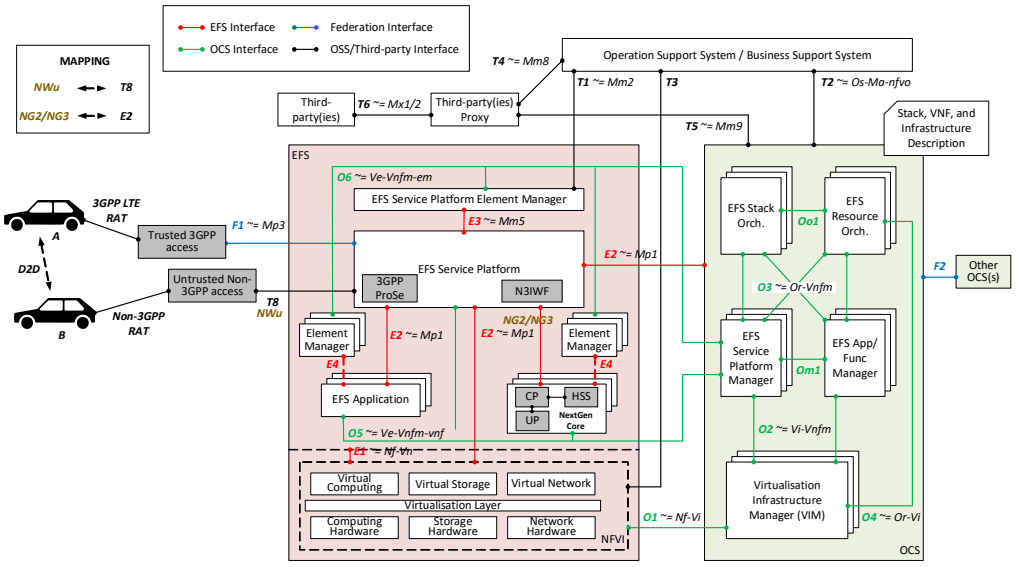

FIGURE 4-16: 5G-CORAL ARCHITECTURE AND MAPPED NON-3GPP ACCESS INTERFACES

4.3.4 Connecting via Bluetooth and ZigBee

Due to the raising and diverging needs of IoT applications, an EFS may exist in an IoT network. An IoT network is usually formed through personal area network (PAN) techniques. The state of art of such techniques are Bluetooth and Zigbee. We investigate the ways to discover EFSs via Bluetooth or Zigbee in the following paragraphs.

4.3.4.1 Connecting via Bluetooth

Discovering the EFS via Bluetooth [42] interfaces can be done via SDP protocol which was specified by the Bluetooth SIG in December 2016. SDP provides a means for SDP clients to discover the existence of services provided by SDP servers. To discover a service, a SDP client issues a SDP request to a SDP server to retrieve service information of the service. A service is described by a service record, which contains a list of attributes to characterise the service.

To carry out effective and efficient service discovery, two main operations are specified: searching and browsing. When a user is able to identify some desired characteristics of a target service, the searching operation is suitable for service discovery. To search for services, service search pattern is used, which is a list of UUIDs. A service search pattern matches a service record if the set of UUIDs in the service search pattern is a subset of UUIDs in the service record. Once a search matches, SDP client receives a service record handle. This handle can be used to request the values of specific attributes of the service record.

To correctly identify an EFS enabled service via Service Searching, a discoverable attribute, say with attribute name to be EFS information service, should be specified with a value of a UUID. In addition, we propose to add two attributes shown in the following table. These two attributes are applicable via Service Searching if a service provided by a Bluetooth device is able to provide internet access.

TABLE 4-8: EFS-SPECIFIC ATTRIBUTE DEFINITIONS

Attribute Name	Attribute ID	Description
EFS URI	0x310	Provides a list of one or more URI for uniquely identifying the EFS available on the network
OCS Entry Point URL	0x311	Provides a URL for the Orchestration and Control System which can be used as entry point for integrating the resource into the existing EFS

To discover an EFS enabled service via Service Browsing, a new service class is specified in the Service Browsing Hierarchy shown in the following table.

TABLE 4-9: EXAMPLE OF SERVICE BROWSING HIERARCHY WITH EFS & OCS ACCESS SERVICE

Service Name	Service Class	Attribute Name	Attribute Value
Reference	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		Group ID	ReferenceID
EFS & OCS Access	ReferenceID	BrowseGroupList	ReferenceID
		EFS URL	EFS access URL
		OCS Entry Point URL	OCS access URL

Please refer to Appendix A for more details about how to discover EFS via SDP.

4.3.4.2 Connecting via Zigbee

Discovering the EFS via Zigbee [43] interfaces can be done via Zigbee Application Layer which includes three main components: Zigbee Application Support (APS) Sub-layer, Zigbee Application Framework (ZAF) and Zigbee Device Object (ZDO). APS provides a means to manage applications. ZAF specifies a way to describe applications systematically. ZDO provides a means to discover devices and services via descriptors defined by ZAF.

A Zigbee node with EFS resource needs to discover a service providing both Internet access and EFS and OCS access information. Taking the Home Automation scenario as an example, the Zigbee node can discover a Zigbee node providing Internet access by specifying the simple descriptor with profile ID 0x0104 (home automation) and with a device ID 0x0050 (home gateway). However, current Zigbee specifications lack the consideration of EFS and OCS design. Therefore, to enable discovery of EFS and OCS access information, we propose to add the two fields to Complex descriptor as shown in the following table. During service discovery, the node descriptor in a discovery response should set the complex descriptor available indicator so as to guide the requesting Zigbee node to further discover the complex descriptor.

TABLE 4-10: PROPOSED EFS-ENABLED FIELDS OF THE COMPLEX DESCRIPTOR

Field Name	XML Tag	Data Type
EFS URL	<efsURL>	Character string
OCS Entry Point URL	<ocsEntryPointURL>	Character string

The method mentioned above merely provides a way to enable an EFS resource to access EFS and/or OCS. For more details, please refer to Appendix A.

4.4 Integrating EFS resources

After having discovered the presence of an EFS, a resource can start the process of integrating its capabilities into the local available system. As introduced in Section 3.5, we further classify the resources as trusted and untrusted based on their ownership. Such differentiation is required for the integration process since untrusted resources require the verification of SLAs against third-party organisations. Figure 4-17 shows the high-level mapping of the integration process onto 5G-Coral architecture.

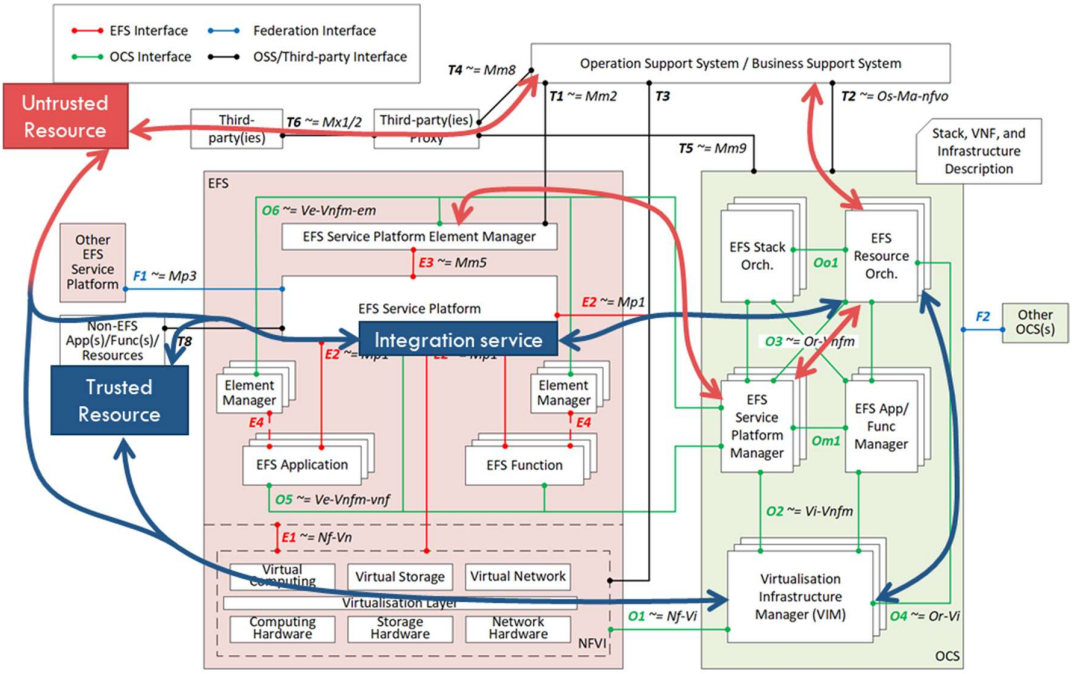

FIGURE 4-17: EFS INTEGRATION APPROACH FOR TRUSTED AND UNTRUSTED RESOURCES

As it can be noticed, untrusted resources interact with the Operation Support System (OSS)/Business Support System (BSS) via the Third-party(ies) Proxy in order to be integrated into the target EFS. Such interaction, highlighted in red, allows an on-demand federation of resources where SLAs can be checked before integration, including pricing and billing agreements. Indeed, such business-related information is available only at OSS/BSS level and not inside the OCS whose goal is to orchestrate services and resources. Figure 4-18 shows the high-level message flow for the initial federation procedure.

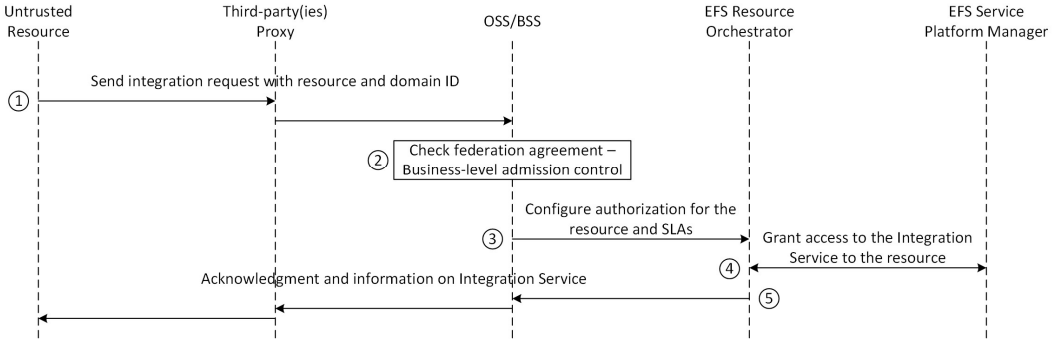

FIGURE 4-18: UNTRUSTED RESOURCE INITIAL FEDERATION PROCEDURE

After having retrieved the OCS Entry Point URL according to the procedures defined in Section 4.3, the **untrusted resources** perform the following steps as depicted in Figure 4-18:

1. Contact the Third-party(ies) via the T6 interface and send an integration request containing its identity including its domain ID. The request is then forwarded to the OSS/BSS via the T4 interface;
2. The OSS/BSS checks the eligibility of the resource for integration into the domain by verifying whether a federation agreement is in place with the third-party domain and whether the resource can be successfully authenticated against this external domain;

3. In affirmative case, the OSS/BSS instructs via the T2 interface the EFS Resource Orchestrator about the eligibility for integration of the untrusted resource and provides the SLA associated to the resource (e.g., privacy, fencing, etc.);
4. The EFS Resource Orchestrator instructs via the O3 interface the EFS Service Platform Manager as to grant to the resource access to the Integration Service (see next paragraph);
5. The EFS Resource Orchestrator informs via the T2 interface the OSS/BSS about the Integration Service details. In turn, the OSS/BSS communicates such details to the resource via the T4 and T6 interface.

Given the dynamicity of the 5G-Coral environment, and the presence of an EFS Service Platform capable of providing information delivery in such environment, we leverage such platform and we define an Integration Service for enabling the communication between the integrating resource and the OCS Resource Orchestrator which keeps track of all the resources and VIMs available in a given domain. This service is used by both trusted and untrusted resources. Indeed, the steps described above are required for enabling access to such service to untrusted resources. Trusted resources instead do not to perform such initial steps since they belong to the same owner operating the infrastructure and are hence authorised during the infrastructure deployment phase. Figure 4-19: shows the high-level message flow for the integration procedure.

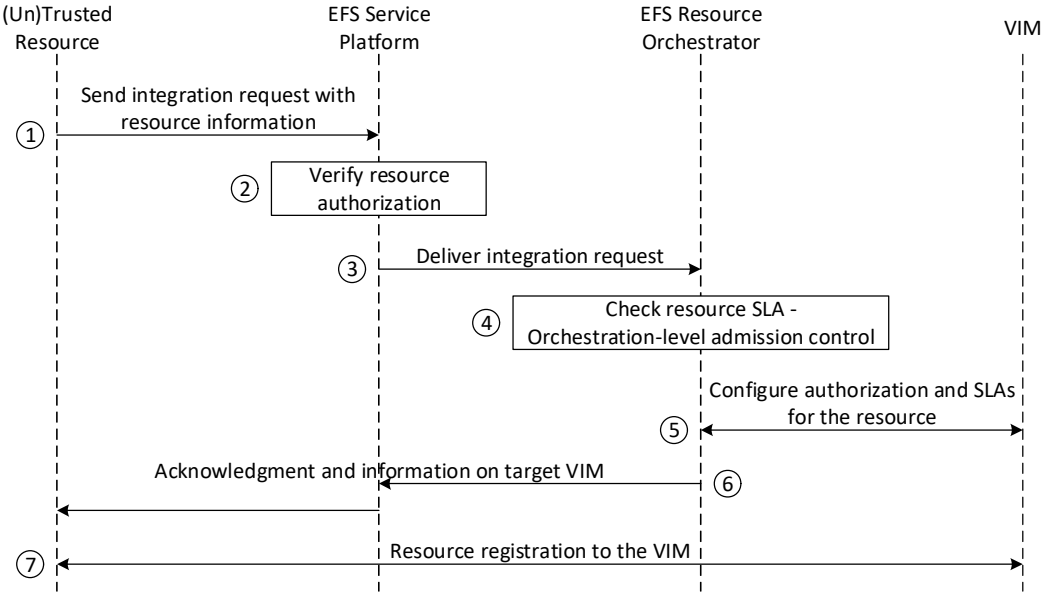

FIGURE 4-19: EFS RESOURCE INTEGRATION PROCEDURE

With the EFS URI information retrieved according to the procedures defined in Section 4.3, the **EFS resources** perform the following steps as depicted in Figure 4-19: :

1. The resources send an integration request via the T8 interface to the Integration Service containing its identity, the domain ID, ant the hardware and software information reported in Table 4-11 and Table 4-12, respectively;
2. The EFS Service Platform verifies that the resource is authorised to access the Integration Service;
3. In affirmative case, the EFS Service Platform delivers the request to the EFS Resource Orchestrator via the E2 interface;
4. The EFS Resource Orchestrator checks the resource SLAs and performs an admission control based on the actual status of the system (e.g., EFS and OCS load);
5. In affirmative case, the EFS Resource Orchestrator instructs the VIM via the O4 interface to authorise integration requests coming from the target resource;

6. The EFS Resource Orchestrator informs via the E2 interface, and next via the T8 interface, the resource about the target VIM it needs to associate;
7. The resource registers to the target VIM and becomes part of the EFS via the O1 interface.

TABLE 4-11: EFS RESOURCE – HARDWARE-RELATED INFORMATION

Type	Parameters
CPU	Architecture, Frequency, Number of cores, CPU extensions, Virtualisation support, CPU model name, Cache size, Quick Path Interconnect (QPI) speed, Turbo-boost enabled, hyperthreading, Thermal Design Power (TDP).
RAM	Amount, Type (e.g. SRAM, DRAM), Frequency.
Network	Interface names, Interface type (Wi-Fi, Ethernet, 4G, 5G, Bluetooth, ZigBee, etc.), Interface parameters (e.g., speed, bandwidth, frequency, channel, transmission modes, etc.), Interface interconnection (e.g., PCI, USB, etc.), MAC addresses, Routing tables, Default gateway, Virtualisation support.
Storage	Storage Type (e.g., SSD, Hard Disk, SD cards), Available space, Mount point (e.g., /home), Filesystem, Storage interconnection (e.g., PCI, Sata, USB, etc.), RAID configuration, Read/Write speed.
I/O devices	Device name, Device type (e.g., GPIO, COM port, etc.), Device path.
HW accelerators	Accelerator type (e.g., GPU, Hardware-based encryption module, etc.), Hardware address (e.g. PCI address), Accelerator name, Supported libraries (e.g., cuda, fpga, netfpga, opencl, etc.).
Sensors	Sensor type (e.g., GPS, temperature, accelerometer, etc.), Sensor address, Sensor name, Supported libraries.

TABLE 4-12: EFS RESOURCE – SOFTWARE-RELATED INFORMATION

TYPE	PARAMETERS
Operating system	Operating system, Version, Release, Architecture (e.g., 32bit vs 64bit), Kernel.
Virtualisation environment	Virtualisation type (e.g., Docker, KVM, LXD, etc.), Virtualisation environment release, Virtualisation environment configuration.
Protocols and runtime	Communication protocols (e.g. DDS, MQTT, etc.), Runtime available (e.g., Python, Java, Ruby, etc.).

5 Conclusions and next steps

WP3 aims at designing an Orchestration and Control system (OCS) for dynamic federation and optimised allocation of 5G-CORAL EFS resources. The initial design of the OCS has been performed during the first nine months of the 5G-Coral project addressing relevant aspects like dynamicity of the resources, discovery, and integration of resources, federation, and monitoring. Particularly, we extended existing industrial frameworks for NFV, MEC, and fog to best suit dynamic environments where EFS resources are volatile. To this end, we extended the ETSI NFV and MEC descriptors into an EFS Stack descriptor capable of expressing heterogeneity, mobility, and volatility of resources along with a service-oriented communication view. Discovery protocols and mechanisms for different RATs have been analysed and extended, thus enabling the discovery of EFS availability and the required information for integration and federation of resources into the target EFS. Finally, we started the design of orchestration and control algorithms for elastic placement and migration of EFS functions and optimised allocation of EFS resources, including federation aspects like pricing.

In the meanwhile, the development of an OCS prototype has started under the name of fog05, and an initial open-source release is available on GitHub[5]. fog05 adopts the plugin-based architecture described in Section 3 and defines the atomic entity abstraction to unify the compute, storage and communication fabric end-to-end and thus allow applications and functions to be managed, monitored and orchestrated across the cloud-to-thing continuum. fog05 implements the Finite State Machine defined in Section 3.2 for managing the life-cycle of applications and functions while encompassing several types of atomic entity, ranging from VMs to containers, Unikernel, and binary executable. Moreover, fog05 adopts the EFS Stack descriptor defined in Section 3.1 as to also define deployment affinity for functions and applications with regard to compute, storage, I/O, and accelerators like GPUs and FPGAs. Finally, fog05 adopts DDS [37]/Zenoh [38] as middleware to for the control and management plane, thus allowing OCS components to run on very different nodes, both in terms of computing and networking capabilities. At the time of writing this document, fog05 implements part of the VIM and EFS Manager functionalities envisioned in the OCS.

In the remaining twelve months of WP3 lifetime, we envision two main tracks: (i) refining the OCS design, including a more detailed specification of OCS components and protocols, and (ii) progressing the development of fog05 and perform a performance assessment of such prototype. Regarding the former track, we plan to specify interfaces to integrate the EFS with external entities (e.g., central clouds) as to enable instantiation and migration of virtual functions and applications. This will involve additional work on the descriptors. Furthermore, seamless migration of functions within the EFS, triggered by context changes or network optimisation requests, will be investigated. Finally, we plan to extend orchestration and control algorithms for elastic placement and migration of EFS functions, and optimised allocation of EFS resources. For what concerns the second track, we plan to extend the fog05 prototype as to also integrate and interact with an Orchestrator and external clouds as to enable instantiation and migration of applications and functions from the EFS to the cloud and vice-versa. Ultimately, we plan to assess the performance of the OCS prototype in terms of scalability (e.g., number of nodes), impact on EFS resources, resiliency to connectivity errors, and application and function migration.

All the findings in this deliverable have already been input to the ongoing architectural refinement in Work Package 1 and EFS design in Work Package 2. Additional input related to federation are also been fed into Work Package 1 as to foster discussion on business perspectives of the various actors envisioned in the 5G-CORAL ecosystem. Work Package 3 is now working closely with Work Package 4 as to progress the development of the initial OCS prototype towards the expected validation and trials.

[5] https://github.com/atolab/fog05

Bibliography

[1] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration," ETSI GS NFV-MAN 001 V1.1.1, Dec. 2012

[2] ETSI GS NFV-IFA 014 V2.4.1 (2018-02)

[3] 3GPP, "Feasibility Study on New Services and Markets Technology Enablers," 3rd Generation Partnership Project, TR 22.891, Sep 2016.

[4] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration; Architectural Options," European Telecommunications Standards Institute, GS NFV-IFA 009, July 2016.

[5] ETSI, "Mobile Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, GS MEC 003, Mar 2016.

[6] OpenFog Consortium, "OpenFog Reference Architecture for Fog Computing," Architecture Working Group, Feb 2017.

[7] 5G-Coral project, "Deliverable D1.1; 5G-CORAL initial system design, use cases, and requirements," Feb. 2018.

[8] C. Simon et al., "5G exchange for inter-domain resource sharing," 2016 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN), Rome, 2016, pp. 1-6.

[9] T. Kurze et al., "Cloud federation," Proceedings of the 2nd International Conference on Cloud Computing, GRIDs, and Virtualization, CLOUD COMPUTING 2011, 2011

[10] R. Moreno-Vozmediano et al., "IaaS cloud architecture: From virtualized datacenters to federated cloud infrastructures," IEEE Computer, vol. 45, no. 12, pp. 65–72, Dec. 2012

[11] Taleb T. et al., "Follow-me cloud: When cloud services follow mobile users," IEEE Transactions on Cloud Computing. 2016.

[12] Panarello A. et al., "Automating the Deployment of Multi-Cloud Applications in Federated Cloud Environments," 10th VALUETOOLS 2017.

[13] ETSI, "Network Functions Virtualisation (NFV); Management and Orchestration; Report on architecture options to support multiple administrative domains," European Telecommunications Standards Institute, DGR NFV-IFA 028, Dec. 2017.

[14] Yuqian Lu et al., "Development of a hybrid manufacturing cloud," Journal of Manufacturing Systems, Oct. 2014.

[15] Jin Li et al., "A hybrid cloud approach for secure authorized deduplication," IEEE Trans. On Parallel and Distributed Systems, 26(5):1206-1216, May 2015.

[16] Yaron Y. Goland et al., "Simple Service Discovery Protocol/1.0; Operating without an Arbiter," IETF Internet Draft, draft-cai-ssdp-v1-03, April 2000.

[17] IANA, "Service Name and Transport Protocol Port Number Registry," [Online]. Available: IANA, https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xml. [Accessed 18th January 2018].

[18] IANA, "Ipv4 Multicast Address Space Registry," [Online]. Available: IANA, https://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml. [Accessed 18th January 2018].

[19] B. Haberman, "Allocation Guidelines for Ipv6 Multicast Addresses," IETF RFC 3307, August 2002.

[20] H. Frystyk Nielsen, "HTTP Extension Framework," IETF Internet Draft, draft-frystyk-http-extensions-03, March 1999.

[21] S. Reddy et al., "DAV Searching & Locating," IETF Internet Draft, draft-ietf-dasl-protocol-00, June 1999.

[22] UPnP Forum, "UPnP Device Architecture 2.0," February 2015.

[23] J. Cohen et al., "General Event Notification Architecture Base," IETF Internet Draft, draft-cohen-gena-p-base-01, July 1998

[24] IEEE 802.11-2016 Standard, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications," December 2016.

[25] Wi-Fi Alliance, P2P Technical Group, "Wi-Fi Peer-to-Peer (P2P) Technical Specification v1.7," July 2016.

[26] D. Camps-Mur, A. Garcia-Saavedra and P. Serrano, "Device-to-device communications with Wi-Fi Direct: overview and experimentation," in IEEE Wireless Communications, vol. 20, no. 3, pp. 96-104, June 2013.
[27] IEEE 802.11u-2011 Amendment, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications Amendment 9: Interworking with External Networks," February 2011
[28] IETF RFC 2461, "Neighbor Discovery for IP Version 6 (Ipv6)," December 1998.
[29] IETF RFC 6763, "DNS-based service discovery", [Online]. Available: IETF, https://tools.ietf.org/html/rfc6763 [Accessed 22nd May 2018].
[30] IETF RFC 2782, "A DNS RR for specifying the location of services (DNS SRV)", 2000 [Online]. Available: IETF, https://www.ietf.org/rfc/rfc2782.txt [Accessed 22nd May 2018].
[31] IETF RFC 1035, "DOMAIN NAMES – IMPLEMENTATION AND SPECIFICATION", 1987 [Online]. Available: IETF, https://www.ietf.org/rfc/rfc1035.txt [Accessed 22nd May 2018].
[32] TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0
[33] 5G CORAL Project, "D2.1 – Initial design of 5G-CORAL edge and fog computing system," June 2018.
[34] 5G CORAL Project, "D1.1 – 5G-CORAL Initial system design, use cases, and requirements," March 2018.
[35] TOSCA Simple Profile in YAML Version 1.2. Edited by Matt Rutkowski, Luc Boutier, and Chris Lauwers. 07 December 2017. OASIS Committee Specification Draft 02 / Public Review Draft 01
[36] "Azure Cosmos DB". Microsoft Azure. Microsoft. Retrieved 9 July2017.
[37] Data Distribution Service. Version 1.4. March 2015. Object Management Group [Online]. Available: https://www.omg.org/spec/DDS/1.4/ [Accessed 22nd May 2018].
[38] Zenoh. Zero Network Over-Head. Version 0.1.8. Angelo Corsaro, Erik Boasson [Online]. Available: http://zenoh.io/download/pdf/2018.04.23-zenoh.pdf [Accessed 22nd May 2018].
[39] ETSI, "Mobile Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, GS MEC 001, Mar 2016.
[40] ETSI, "Mobile Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, GS MEC 011, Jul 2017.
[41] ETSI, "Mobile Edge Computing (MEC); Framework and Reference Architecture," European Telecommunications Standards Institute, GS MEC 010-2, Oct 2017.
[42] Bluetooth Special Interest Group [Online]. Available: https://www.bluetooth.com/ [Accessed 22nd May 2018].
[43] Zigbee Alliance [Online]. Available: https://www.zigbee.org/ [Accessed 22nd May 2018].
[44] Misbah Liaqat, Victor Chang, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Toseef, Umar Shoaib, Rana Liaqat Ali, Federated cloud resource management: Review and discussion, Journal of Network and Computer Applications, Volume 77, 2017, Pages 87-105, ISSN 1084-8045.
[45] M. Mihailescu and Y. M. Teo, "Dynamic Resource Pricing on Federated Clouds," 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 2010, pp. 513-517. Doi: 10.1109/CCGRID.2010.123.
[46] M. Mihailescu and Y. M. Teo, "Dynamic Resource Pricing on Federated Clouds," 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing, Melbourne, Australia, 2010, pp. 513-517. Doi: 10.1109/CCGRID.2010.123.
[47] Qanbari, S., Li, F., Dustdar, S., Dai, T., 2014. Cloud asset pricing tree (CAPT): elastic economic model for cloud service providers. Chinacloud, 221–229.
[48] Misbah Liaqat, Victor Chang, Abdullah Gani, Siti Hafizah Ab Hamid, Muhammad Toseef, Umar Shoaib, Rana Liaqat Ali, Federated cloud resource management: Review and discussion, Journal of Network and Computer Applications, Volume 77, 2017, Pages 87-105, ISSN 1084-8045.
[49] Fan, T., Liu, J., Gao, F., 2013. Dynamic Pricing Trade Market for Shared Resources in IIU Federated Cloud. Vol. 22, pp. 170–175.

[**50**] Hassan M, Song B, Huh EN (2011) Game-based distributed resource allocation in horizontal dynamic cloud federation platform. In: Xiang Y, Cuzzocrea A, Hobbs M, Zhou W (eds) Algorithms and Architectures for Parallel Processing, vol 7016., Lecture Notes in Computer ScienceSpringer, New York, pp 194–205.
[**51**] IEEE 802.11u, table 7-42BH
[**52**] IEEE 802.11u, table 7-25M
[**53**] Yaron Y. Goland et al., "Simple Service Discovery Protocol/1.0; Operating without an Arbiter," IETF Internet Draft, draft-cai-ssdp-v1-03, April 2000.
[**54**] B. Haberman, "Allocation Guidelines for Ipv6 Multicast Addresses," IETF RFC 3307, August 2002.
[**55**] L. Cominardi, O.I. Abdullaziz, K. Antevski, S.B. Chundrigar, R. Gdowski, P.-H. Kuo, A. Mourad, L.-H. Yen, and A. Zabala, "Opportunities and Challenges of Joint Edge and Fog Orchestration," in IEEE WCNCW - Compass 2018: 2018 IEEE Wireless Communications and Networking Conference Workshops (WCNCW): The First Workshop on Control and management of Vertical slicing including the Edge and Fog Systems (COMPASS), Barcelona, April 2018.
[**56**] CJ. Bernardos, A. Rahman, and A. Mourad, "Service Function Chaining Use Cases in Fog RAN," IETF Internet Draft, draft-bernardos-sfc-fog-ran-03, March 2018
[**57**] CJ. Bernardos and A. Mourad, "Service Function discovery in fog environments," IETF Internet Draft, draft-bernardos-sfc-discovery-00, March 2018
[**58**] CJ. Bernardos and A. Mourad, "IPv6-based discovery and association of Virtualization Infrastructure Manager (VIM) and Network Function Virtualization Orchestrator (NFVO)," IETF Internet Draft, draft-bernardos-nfvrg-vim-discovery-00, March 2018
[**59**] fog05 [Online]. Available: https://github.com/atolab/fog05 [Accessed 22nd May 2018].
[**60**] 3GPP, "Policy and Charging Control Framework for the 5G System", TS 23.503 V15.1.0

Appendix A: Discovery Protocols

A significant challenge for constrained wireless communications is to make devices as autonomous as possible such that once deployed, they become invisible. For this purpose, low power devices should require minimal human intervention at every stage of their operating life. This is important for large networks where configuration and maintenance of hundreds of devices often becomes a significant challenge, if not a burden, in particular when wireless devices join or leave the network while they move around the environment and/or their connectivity changes.

The traditional approach to reduce configuration and administration of network devices is with the help of service discovery mechanism.

A.1 Simple Service Discovery Protocol

The Simple Service Discovery Protocol (SSDP) is a text-based network protocol for advertisement and discovery of HTTP clients and HTTP resources in local area networks [53]. It accomplishes this without assistance of server-based configuration mechanisms, such as the Dynamic Host Configuration Protocol (DHCP) or the Domain Name System (DNS), and without special static configuration of a network host. Indeed, SSDP uses the User Datagram Protocol (UDP) as underlying transport protocol for HTTP messages. SSDP uses the IANA-assigned port number 1900 [17]. Additionally, SSDP relies on IP multicast to deliver those messages over the network. In Ipv4, the multicast address is "239.255.255.250" [18], while in Ipv6 the multicast address depends on the scope: "ff02::c" for Ipv6 link-local, "ff05::c" for site-local, "ff08::c" for organisation-local, and "ff0e::c" for Ipv6-global [54].

SSDP considers two types of requests to be sent across the SSDP multicast channel/port: the first are discovery requests sent by SSDP clients looking for particular SSDP services, the second instead are presence announcements sent by SSDP services announcing their presence. To discover SSDP services in the network, an SSDP client may perform active discovery by sending on-demand requests to the SSDP multicast address/port. Specifically, the client uses the HTTP method M-SEARCH for the discovery request[6]. If a SSDP service hears a request that matches the service it offers, then it will respond to the client via a unicast HTTP UDP response. Likewise, SSDP services may send announcements to the SSDP multicast channel/port to advertise their presence in the network. In this way, SSDP clients can passively discover the available services by periodically receiving the service announcements. SSDP uses the HTTP method NOTIFY to announce the establishment or withdrawal of services (presence) information to the multicast group.

Services are identified by a unique pairing of a service type Uniform Resource Identifiers (URI) and a Unique Service Name (USN). Service types identify a type of service, such as a printer, shared folder, etc. The exact meaning of a service type is outside the scope of SSDP which only considers a service type as an opaque identifier that identifies a particular type of service. A USN is a URI that uniquely identifies a particular instance of a service. USNs are used to differentiate between two services with the same service type. In addition to providing both a service type and a USN, discovery results and presence announcements also provide expiration and location information. Location information identifies how a client should contact a particular service. One or more location URIs may be included in a discovery response or a presence announcement. Expiration information identifies how long a SSDP client should keep information about the service in its cache. Once the entry has expired it is to be removed from the SSDP client's cache.

A.2 Universal Plug and Play

Universal Plug and Play (UpnP) is a set of networking protocols for residential networks that permits networked devices to seamlessly discover each other's presence on the network and establish functional network services for data sharing, communications, and entertainment. UPnP assumes the network runs Internet Protocol (IP) and then leverages HTTP, SOAP and XML on top of IP, in order to provide device/service description, actions, data transfer and eventing. To arbitrate such interaction, UPnP categorise the devices in two types: control points (CPs) and controlled devices

[6] The SSDP M-SEARCH method is an HTTP extension [20] of the HTTP SEARCH method [21].

(CDs), with the former controlling the latter [7]. To allow those devices to discover each other, UPnP leverages the SSDP protocol [1]. When a device is added to the network, the device advertises its services to the control points on the network via SSDP messages. After a control point has discovered a device, the control point retrieves the device's description from the location (URL) provided by the device in the discovery message.

The UPnP device description includes a list of any embedded services and vendor-specific information like the model name and number, serial number, etc. For each service, the device description lists the URLs for control, eventing and the commands/actions to which the service responds. The description for a service also includes a list of variables which model the state of the service at run time. As a result, the control point can send actions to a device's service according to its description and the effects of the action, if any, are modelled by changes in the service variables. Upon variable changes, the services may publish updates and a control point may subscribe to receive this information. Such event notification system is based on the General Event Notification Architecture (GENA) [8] and it is usually implemented in residential networks via a single control point. To support scenarios with multiple control points, eventing is designed to keep all control points equally informed about the effects of any action. Therefore, all subscribers are sent all event messages, subscribers receive event messages for all the updated variables, and event messages are sent no matter why the state variable changed. This approach is clearly unsuitable in scenarios larger than traditional residential networks.

A.3 Wi-Fi Discovery

Conventional Wi-Fi networks are typically based on the presence of controller devices known as wireless access points. The majority of Wi-Fi networks are configured in *infrastructure mode* [24], where the access point acts as a central hub and any communications of the connected devices goes through it. Direct device to device connectivity was already possible in the original IEEE 802.11 standard by means of the *ad-hoc mode* of operation. However, the lack of additional information for discovery makes this operation mode difficult to use in practice. At that end, Wi-Fi Direct, initially called Wi-Fi P2P, is a Wi-Fi standard enabling devices to easily connect with each other without requiring a wireless access point [25]. A salient feature of Wi-Fi Direct is the ability to support service discovery at the link layer. In this way, prior to the establishment of the D2D connection, devices can exchange queries to discover the set of available services and, based on this, decide whether to continue with the formation of a P2P group or not [26]. Notice that this represents a significant shift from traditional Wi-Fi networks, where it is assumed that the only service clients are interested in is Internet connectivity.

In order to implement the above, IEEE defined in February 2011 the IEEE 802.11u protocol and is currently working on the 802.11aq amendment which extends some of the discovery functionalities. One of the motivations for this standard is to allow Wi-Fi client devices to learn more about a network before deciding to join it. IEEE 802.11u/aq allow indeed mobile devices to gain additional insight on the network capabilities to perform a better network selection, automated roaming and offload, secure authentication, and QoS integration with operator networks carrying user traffic. Those devices can use active (probes) or passive (beacons) scanning techniques to discover surrounding Access Points (APs), learn about the network, determine which network is best, and make a connection. Traditionally, this process depends on device recognition of the network name: the SSID. 802.11u/aq do not fundamentally alter the basic discovery process; however, they do enable the discovery of additional information during the scanning process and it allows the client to query the AP for additional information. Each beacon or probe response carries information about the AP's capabilities in a component of the frame called an information element. The 802.11u/aq protocol focuses on enhancing network discovery by adding additional information elements to these frames as reported in Table A-1.

TABLE A-1: IEEE 802.11U INFORMATION ELEMENTS

Information Element Name	Description
Extended Capabilities	Indicates whether an AP supports 802.11u interworking features.
Interworking	Identifies the interworking capabilities of the AP or device.

Roaming Consortium	Identifies service providers or groups of roaming partners whose security credentials can be used to connect to a network.
Advertisement Protocol	Identifies the network's support for particular advertisement protocols, such as ANQP, which allow the client to learn more about the network by querying the AP prior to forming a connection.

The **Extended Capabilities Element** existed prior to 802.11u/aq, however some of the previously unused bits are now used to indicate AP support for interworking features:

- Interworking – Bit is set (to 1) to indicate interworking support;
- QoS Map – Bit set to indicate support for QoS mapping from 802.11 to external networks;
- SSPN Interface – Bit set to indicate that the AP has a logical backend interface to external service providers (SSPN – Subscription Service Provider Network);

The **Interworking Element** communicates basic features related to interworking services. AP or devices advertise the support of 802.11u/aq protocol by including this element in their frames. APs include the Interworking element in beacons and probe responses and devices include it in probe requests and (re)association requests. The Interworking Element provides additional information such as access network type, venue information, and homogeneous extended service set (HESSID).

TABLE A-2: ACCESS NETWORK TYPE [51]

Access net. type	Meaning	Description
0	Private network	Non-authorised users are not permitted on this network. Examples of this access network type are home networks and enterprise networks, which may employ user accounts. Private networks do not necessarily employ encryption.
1	Private network with guest access	Private network but guest accounts are available. Example of this access network type is enterprise network offering access to guest users.
2	Chargeable public network	The network is accessible to anyone, however, access to the network requires payment. Further information on types of charges may be available through other methods (e.g., IEEE 802.21, http/https redirect or DNS redirection). Examples of this access network type is a hotspot in a coffee shop offering Internet access on a subscription basis or a hotel offering in-room Internet access service for a fee.
3	Free public network	Identifies service providers or groups of roaming partners whose security credentials can be used to connect to a network.
4	Personal device network	A network of personal devices. An example of this type of network is a camera attaching to a printer, thereby forming a network for the purpose of printing pictures.
5	Emergency services only network	A network dedicated and limited to accessing emergency services.
6-13	Reserved	Reserved.
14	Test or experimental	The network is used for test or experimental purposes only.
15	Wildcard	Wildcard access network type.

Table A-2 reports the distinct types of access network as defined in [24], while the venue group codes are reported in Table A-3. Please note that the standard IEEE 802.11u/aq defines a large set of codes for different venue types (see Table 7-25n in [24]) which it is not reported in this document for sake of brevity. Normally, including venue information in the Interworking element

may be used to improve network selection behaviour by devices. Users or operators may be able to more precisely control network connection preferences by using venue types in the decision criteria. Mobile devices can also adopt venue-specific device behaviours based on the advertised information (e.g., enabling Wi-Fi offloading when in a train).

TABLE A-3: VENUE GROUP CODES AND DESCRIPTIONS [52]

Venue Group code	Venue Group description	Venue Type code example	Venue Type description example
0	Unspecified	--	--
1	Assembly	3	Passenger terminal (e.g., train station)
2	Business	8	Bank
3	Educational	3	University or College
4	Factory and Industrial	1	Factory
5	Institutional	1	Hospital
6	Mercantile	4	Shopping Mall
7	Residential	2	Hotel or Motel
8	Storage	1-255	Reserved
9	Utility and Miscellaneous	1-255	Reserved
10	Vehicular	6	Train
11	Outdoor	4	Traffic Control
12-255	Reserved	--	--

A **homogenous extended service set** (ESS) is a group of basic service sets that all provide access to the same external networks. A homogenous ESS is identified by its HESSID, which takes the form of a MAC address (6 bytes) and it is globally unique. As such, it could be used in concert with the SSID to identify a specific service provider network. Devices could also use the HESSID value to ensure that a mobility event (e.g., Wi-Fi roaming) would not disrupt a specific application service. The device would know if the new service set is in the same homogenous ESS as its existing connection. This enables the creation of a **roaming consortium** formed by a group of service providers (SP) where the same user's credentials can be used for authentication. Service providers in a roaming consortium have user roaming agreements with one another. As a result, the **Roaming Consortium Element** tells a mobile device which roaming consortiums or service providers are available through some AP. Roaming consortia are identified by an organisation identifier (OI) that is assigned by the IEEE and it is globally unique. During network discovery and selection, the devices will receive this list of OIs and determine if any of them meet the device's connection and roaming policies. The Roaming Consortium element can include up to 3 OIs in the beacon but will also notify clients if additional OIs are available.

TABLE A-4: COMMON ADVERTISEMENT GROUP INFORMATION TYPE DEFINITIONS

Value	Name
0	Access Network Query Protocol (ANQP) – *Support for this protocol is mandatory*
1	Media Independent Handover (MIH) Information Service
2	Media Independent Handover (MIH) Command and Event Services Capability Discovery
3	Emergency Alert System
4	Registered location query protocol (RLQP)
5-127	Reserved
128	Access Network Query Protocol (ANQP) with Service Information Registry
129-220	Reserved
221	Vendor Specific
222-255	Reserved

To prevent airtime saturation with AP service advertisements, a subset of interworking information is advertised in beacons, but the remaining information is provided to clients only by request.

Indeed, some or all of the interworking information will be irrelevant to some devices, especially those that do not support 802.11u. Therefore, The AP communicates the most vital information to all client stations, and then advertises a way for clients to individually discover more, if desired. The **Advertisement Protocol Element** facilitates this function by identifying the advertisement protocol(s) by which a client may query the AP for more information as shown in Table A-4. When the AP receives such queries from a station, it may either reply directly or proxy the request to an external advertisement Server. The support of Access Network Query Protocol (ANQP) is mandatory in IEEE 802.11u and the information that can be discovered through this protocol are reported in Table A-5.

TABLE A-5: ANQP INFORMATION ID DEFINITIONS

Info ID	Info name	Description
0-255	Reserved	--
256	ANQP Query list	Indicates in a GAS Request a list of elements the device would like to receive in a GAS Response
257	ANQP Capability list	Signals in a GAS Response the ANQP elements supported by the AP
258	Venue Name information	Indicates the name of the venue for the network
259	Emergency Call Number information	Provides a list of location-specific emergency numbers
260	Network Authentication Type information	Indicates the additional steps required for access like on-line enrolment, DNS redirection, acceptance of terms and conditions, HTTP/HTTPS redirection, etc.
261	Roaming Consortium list	Provides the same information as the Roaming Consortium Element in beacons and probe responses but this list is not limited to 3 OIs
262	IP Address Type Availability information	Signals the availability of IP address version and type that could be allocated to the device after successful association (e.g., Ipv4, IPv6, NAT, etc.)
263	NAI Realm list	Identifies all Network Access Identifier realms [ref RFC 4282] for authenticating the users
264	3GPP Cellular Network information	Identifies the 3GPP cellular networks available through the AP
265	AP Geospatial Location	Provides the AP's location in longitude, latitude, and elevation
266	AP Civic Location	Provides the AP's location in civic format (country, province, state, city, street, house number, etc.)
267	AP Location Public Identifier URI	Provides a reference URI at which location information can be retrieved
268	Domain Name list	Lists one or more domain names for the entity operating the AP
269	Emergency Alert Identifier URI	Provides a URI for Emergency Alert System (EAS) message retrieval
270	TDLS Capability	
271	Emergency NAI	Provides an NAI string that the client can use with EAP authentication to access emergency services
272	Neighbour Report	Provides zero or more neighbour reports about neighbouring APs
273-276	Reserved	--
277	Venue URL	Lists one or more URLs which can be used for web page advertising services or providing information particular to a venue
278	Advice of Charge	Provides a list of one or more financial advice of charges related to access to the network

279	Local Content	Provides a list of one or more URLs which can be used to display local content related to the network
280	Network Authentication Type with Timestamp	Provides similar information to that of the Network Authentication Type. A timestamp field indicates the time at which the received terms and conditions were most recently modified. With this timestamp, the requesting devices can determine if previously received information is stale.
281	Service Information Request	Contains a generic request for service information
282	Service Information Response	Contains the detailed service information in response to a Service Information Request
283-56796	Reserved	--
56797	ANQP vendor-specific list	--
56798-65535	Reserved	--

A.4 Service Location Protocol

Service Location Protocol (SLP) [IETF RFC 2608] enables automatic service discovery in an IP-based network without prior configuration. The key role in SLP is the Dictionary Agents (DAs) which is responsible for caching and provision service location information. To collect and cache service location information, a DA announces its existence to Service Agents (SAs) which provide service(s). In response, a SA registers to the DA with the services that it is able to provide. When requesting particular service, a User Agent (UA) first queries the DA for the service location information of the SAs which are able to provide such service. After receiving the information, the UA requests service from that SA.

The location of a DA can be either statically configured in a UA or dynamically obtained through DHCP or the SLP multicast convergence algorithm. The SLP Directory Agent Option is one of the DHCP options with code 78. This option allows a DHCP server to provide a list of IP addresses of DAs. While a UA is acquiring an IP address, it also obtains the IP addresses of the DAs which are available in the network. Alternatively, the SLP multicast convergence algorithm can be used to discover all related DAs in the network. A UA multicast a service request and receives responses from DAs during a wait period. Since service request messages could be lost due to some network condition, it iterates such procedure until no response is received. At DA side, a DA only respond the same request once; therefore, the iteration is convergent. To enable the “once” response, a field called “previous responder list” is introduced and included in the service request message after the first iteration. The presence of a DA in the previous responder list implies that the corresponding UA has been aware of the presence of the DA.

If a DA is not present in a network, a UA uses the SLP multicast convergence algorithm to discovery possible SAs, instead of DAs.

A.5 Web Services Dynamic Discovery

Web Services Dynamic Discovery [OASIS Standard Web Services Dynamic Discovery (WS-Discovery) 1 July 2009] is a specification owned by OASIS that aims to discover and locate services. The protocol defines two modes of operation: ad hoc mode and managed mode.

In the ad hoc mode, a client that wish to find a service by the type or scope in which the service resides, starts by sending a probe message to a multicast group. Those service which matches the prove message, response directly to the client using unicast communication. The client can also find a service by name. It that case, it sends a resolution request message to the multicast group. Again, the service that matches the request will send a unicast response to the client. A new service that arrives at the ad hoc network will send an announcement message to the multicast group in order to alert clients.

If a discovery proxy is available, the managed mode can be set. This alternative mode extends the reach of the protocol beyond the range of an ad hoc network and suppresses multicast communication. The different services will send unicast Hello/Bye messages to the discovery proxy, who will respond to clients on behalf of the services discovered. Thus, clients will send unicast probe/resolve messages to the discovery proxy. The proxy will respond with unicast message if any service matches the client requests.

Clients start in Ad hoc mode until they receive a unicast Hello from a discovery proxy that they trust, when the mode will be managed. If the proxy is not responding, the client will return to the ad hoc mode.

The messages specified are: Hello and Bye; Probe and Probe Match; Resolve and Resolve Match.

A.6 Link Layer Discovery Protocol

Link Layer Discovery Protocol [IEEE 802.1AB-REV – Station and Media Access Control Connectivity Discovery 11 September 2009] allows devices that operate at low layers of the protocol stack (e.g. layer 2 bridges) to learn some of the characteristics of LAN devices available to higher layer protocols. Devices configured with an active LLDP agent send periodically LLDP frames on ports that run the protocol and listen for neighbour messages too. The information gathered is stored in a network device as a management information database (MIB) and is queried with Simple Network Management Protocol (SNMP). Topology information can be obtained by tracking the devices and accessing the MIB.

Regarding LLDP, it allows devices attached to a LAN network to advertise its capabilities to other devices. The transmission is handled by LLDP agent, executed in each LLDP-aware device. The information is stored in the Management Information Base that is accessible using, for instance, SNMP. Aside from the information transmission, the LLDP agent receives information about the capabilities of systems associated with a remote SAP identifier.

The frames transmitted are named Link Layer Discovery Protocol Data Unit (LLDPDU), which carry a set of TLV fields. The information to be transmitted is stored in LLDP local system MIB. The information received, in LLDP remote system MIB. For a transmission, the LLDP entity extracts the objects from the local MIB and generates the TLVs that will be part of the frame. While some of them are already defined, organisations can define its specific TLV.

TABLE A-6: TLV TYPES

Type	Name	Usage	Description
0	End of LLDPDU	Optional	Two bytes, all zero.
1	Chassis ID	Mandatory	Identifies the chassis containing the LAN station associated with the transmitting LLDP agent. A chassis subtype field is part of the TLV.
2	Port ID	Mandatory	Identifies the port component of the MSAP identifier associated with the transmitting LLDP agent. A port subtype indicates how the port is being referenced in the port ID.
3	Time To Live	Mandatory	Time that the information transmitted is valid. If 0, the LLDP agent receiving the frame is notified to delete all information associated with the port.
4	Port description	Optional	Alpha-numeric string
5	System name	Optional	Alpha-numeric string to advertise the assigned name of the system
6	System description	Optional	Alpha-numeric string, textual description of the network entity.
7	System capabilities	Optional	Bit-map defining primary functions of the system. Indicates if the capability is enabled or not too.
8	Management address	Optional	Address of the local LLDP agent, used to reach higher layer entities. This TLV presents many fields. Includes a

			subtype to indicate the type of address. The typical value is a layer 3 IPv4 address.
9-126	Reserved		
127	Organisationally specific TLV	Optional	TLV defined by organisations to advertise information

Regarding the LLDPDU format, the Figure A-1 describes it.

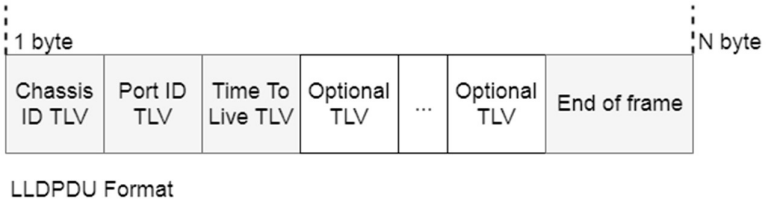

FIGURE A-1: LLDPDU FORMAT

Where, for each TLV, it the format is depicted as follows in Figure A-2.

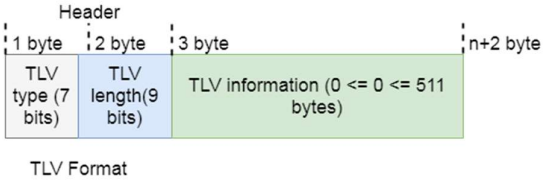

FIGURE A-2: TLV FORMAT

The first time that an access point is installed in the shopping mall and connected to the network that communicates with the EFS, the discovery process must start. Both the wired access point and EFS have to know that an EFS system is available and a new access point is usable. LLDP could support the advertisement of IoT Access Points adjusting the specific TLVs. LLDP frames must be delivered to the EFS devices, in order to collect the information. If the network topology does not allow that, a LLDP proxy is needed to forward the frames to a sink device that will process them.

What features can be discovered in a LAN are listed in the Table A-7.

TABLE A-7: TLV-TYPE TO DISCOVERABLE FEATURES

TLV-Type	Discoverable features
5-6	System name and description
1-2-4	Port name and description
7	VLAN name
8	IP management address
7	System capabilities (switching, routing, etc.)
1-2	MAC/PHY information
7	Link aggregation

These features can be extended thanks to Media Endpoint Discovery, which is an enhancement of LLDP, known as LLDP-MED, providing the following characteristics:

- Auto-discovery of LAN policies (such as VLAN, Layer 2 Priority and Differentiated services settings) enabling plug and play networking;
- Device location discovery to allow creation of location databases and, in the case of Voice over Internet Protocol (VoIP), enhances 911 services;
- Extended and automated power management of Power over Ethernet (PoE) end points;
- Inventory management, allowing network administrators to track their network devices and determine their characteristics.

A.7 DNS-based Service Discovery

DNS-based Service Discovery is a discovery mechanism allowing clients in a given domain to discover a list of named instances of a desired service through standard DNS queries [29]. This solution with existing unicast DNS server and client software as well as Multicast DNS in a zero-configuration environment. DNS SRV [30] and DNS TXT [31] are used to describe each service instance. More specifically, to discover the list of available instances for a given service type, a client queries the DNS PTR record of that service's type name. Then, the server returns zero or more names as pairs "<Service>".<"Domain">, which corresponds to an SRV/TXT record pair. The SRV record is in charge of resolving to the domain name providing the instance, whereas the TXT can include service-specific configuration parameters. Finally, a client can resolve the A/AAAA record for the domain name and connect to the service.

DNS-SD requests can also be sent over a multicast link and, if combined with multicast DNS (mDNS), it can deliver zero-configuration DNS-SD. It still uses DNS PTR, SRV, TXT records to advertise instances of service types, domain names for those instances, and additional configuration parameters for connecting to those instances. However, SRV can now resolve to multicastable.local domain names, which mDNS can resolve to local IP addresses.

DNS-SD is used by Apple products, network printers, several Linux distributions including Debian and Ubuntu, and a number of third-party products for operating systems. For instance, Xerox enterprise printers supports Apple AirPrint via Wide-Area DNS-SD by enabling DNS-SD in the DNS server and registering the DNS-SD records with the DNS server.

A.8 Neighbor Discovery Protocol

The Neighbor Discovery Protocol (NDP) [28] is used with Internet Protocol Version 6 (IPv6) and operates at the link layer in the TCP/IP model. Broadly speaking, NDP is typically used for determining the link-layer addresses for neighbours known to reside on attached links and for quickly purging cached values that become invalid. Moreover, hosts can use NDP to find neighbouring routers that are willing to forward packets on their behalf. Finally, nodes (hosts and routers) use the protocol to actively keep track of which neighbours are reachable and which are not, and to detect changed link-layer addresses, i.e. when a router or the path to a router fails, a host actively searches for functioning alternates.

NDP addresses multiple challenges related to the interaction between nodes attached to the same link, ranging from resource discovery, e.g., router discovery, prefix discovery or parameter discovery such as link MTU or hop limit value, to address autoconfiguration and address resolution, which allows a node to determine the link-layer address of an on-link destination (e.g., a neighbour) given only the destination's IP address. Furthermore, NDP features next-hop determination, which maps an IP destination address into the IP address of the neighbour to which traffic for the destination should be sent, Neighbour Unreachability Detection, Duplicate Address Detection, and Redirect, which specifies how a router informs a host of a better first-hop node to reach a destination.

NDP defines five different ICMP packet types as follows:

1. **Router Solicitation**: When an interface becomes enabled, hosts may send out Router Solicitations that request routers to generate Router Advertisements immediately rather than at their next scheduled time.
2. **Router Advertisement**: Routers advertise their presence together with various link and Internet parameters either periodically, or in response to a Router Solicitation message. Router Advertisements contain prefixes that are used for on-link determination and/or address configuration, a suggested hop limit value, etc.
3. **Neighbour Solicitation**: Sent by a node to determine the link-layer address of a neighbour, or to verify that a neighbour is still reachable via a cached link-layer address. Neighbour Solicitations are also used for Duplicate Address Detection.
4. **Neighbour Advertisement**: A response to a Neighbour Solicitation message. A node may also send unsolicited Neighbour Advertisements to announce a link-layer address change.

5. **Redirect**: Used by routers to inform hosts of a better first hop for a destination.

NDP is also responsible for changing the link-layer address. Specifically, a node that knows its link-layer address has changed can multicast a few unsolicited Neighbour Advertisement packets to all nodes to quickly update cached link-layer addresses that have become invalid. Note that the sending of unsolicited advertisements is a performance enhancement only i.e. unreliable, while the Neighbour Unreachability Detection algorithm ensures that all nodes will reliably discover the new address, though the delay may be somewhat longer.

Furthermore, NDP can perform inbound load balancing, which may occur when nodes with replicated interfaces wish to load balance the reception of incoming packets across multiple network interfaces on the same link. Such nodes have multiple link-layer addresses assigned to the same interface. For example, a single network driver could represent multiple network interface cards as a single logical interface having multiple link-layer addresses. Load balancing is handled by allowing routers to omit the source link-layer address from Router Advertisement packets, thereby forcing neighbours to use Neighbour Solicitation messages to learn link-layer addresses of routers. Returned Neighbour Advertisement messages can then contain link-layer addresses that differ depending on who issued the solicitation.

NDP also introduces anycast addresses to identify one of a set of nodes providing an equivalent service, and multiple nodes on the same link may be configured to recognise the same Anycast address. The protocol handles anycasts by having nodes expect to receive multiple Neighbour Advertisements for the same target. All advertisements for anycast addresses are tagged as being non-Override advertisements. This invokes specific rules to determine which of potentially multiple advertisements should be used.

Finally, NDP includes proxy advertisements. More specifically, a router willing to accept packets on behalf of a target address that is unable to respond to Neighbour Solicitations can issue non-Override Neighbour Advertisements. Proxy advertising could potentially be used to handle cases like mobile nodes that have moved off-link. However, it is not intended as a general mechanism to handle nodes that, e.g., do not implement this protocol.

A.9 Cisco Discovery Protocol

Cisco Discovery Protocol (CDP) is a proprietary Data Link Layer protocol developed by Cisco Systems. It is used to share information about other directly connected Cisco equipment, such as the operating system version and IP address. CDP can also be used for On-Demand Routing, which is a method of including routing information in CDP announcements so that dynamic routing protocols do not need to be used in simple networks.

Cisco devices send CDP announcements to the multicast destination address `01-00-0c-cc-cc-cc`, out each connected network interface. These multicast frames may be received by Cisco switches and other networking devices that support CDP into their connected network interface. This multicast destination is also used in other Cisco protocols such as Virtual Local Area Network (VLAN) Trunking Protocol (VTP). By default, CDP announcements are sent every 60 seconds on interfaces that support Subnetwork Access Protocol(SNAP) headers, including Ethernet, Frame Relay and Asynchronous Transfer Mode (ATM). Each Cisco device that supports CDP stores the information received from other devices in a table that can be viewed using the *show cdp neighbors* command. This table is also accessible via Simple Network Management Protocol (SNMP). The CDP table information is refreshed each time an announcement is received, and the hold-time for that entry is reinitialised. The hold-time specifies the lifetime of an entry in the table - if no announcements are received from a device for a period in excess of the hold-time, the device information is discarded (default 180 seconds).

The information contained in CDP announcements varies by the type of device and the version of the operating system running on it. This information may include the operating system version, hostname, every address (i.e. IP address) from all protocol(s) configured on the port where CDP frame is sent, the port identifier from which the announcement was sent, device type and model, duplex setting, VTP domain, native VLAN, power draw (for Power over

Ethernet devices), and other device specific information. The details contained in these announcements is easily extended due to the use of the type-length-value (TLV) frame format.

A.10 Simple Service Location Protocol

As early as 2005, it started developing a nano-IP stack including a reduced version of HTTP and a directory-less version of SLP that uses WAP Binary XML (WBXML) compression mechanisms for reduced parsing complexity. However, the nanoSLP protocol does not support the use of multicast and has very limited search capabilities. In the same year, the Simple Service Location Protocol (SSLP) for 6LoWPAN also started its standardisation in the IETF Networking Group.

SSLP uses Tokenised XML strings to minimise packet exchanges and adds support for translation agents (TA) to the standard SLP framework. The TAs run on gateways to perform the translation between SLP messages in an IP network and SSLP messages in a 6LoWPAN network. However, this solution involves complexity and incurs delay in translation, each time a message is translated to or from the SLP.

A.11 ZigBee Device Profile

Zigbee device discovery provides the facility for devices to find node-wide (not application/endpoint specific) information about other devices in a network, such as addresses, manufacturer ID, types of applications running, power source, and sleep behaviour. The device discovery is mostly used to learn about device capabilities, in particular for cases where a manufacturer implements extended ZigBee commands. In addition to the device related services, the Zigbee Device Profile (ZDP) also contains a variety of standard mechanisms, called ZDP service discovery, to identify the applications and their properties running on the devices.

Service discovery is performed mainly during device configuration and integration into a ZigBee network by requesting a so-called simple descriptor from other devices of interest which describes everything to know about the endpoint: the Application Profile ID, supported input and output clusters.

Discovering the EFS via Zigbee [43] interfaces can be done via Zigbee Application Layer which includes three main components: Zigbee Application Support (APS) Sub-layer, Zigbee Application Framework (ZAF) and Zigbee Device Object (ZDO). APS provides a means to manage applications. ZAF specifies a way to describe applications systematically. ZDO provides a means to discover devices and services via descriptors defined by ZAF.

A Zigbee node (a single physical Zigbee radio) has 255 endpoints and can support up to 254 applications on endpoints 0x01-0xFE, where the endpoint with value 0x00 is associated with ZOD for device profile information communications and the endpoint 0xFF is for broadcast purpose to address all active endpoints. In a Zigbee node, an application is associated with one and only one endpoint. An application (i.e., service) is described by descriptors as shown in the following table, where the first three descriptors are mandatory.

TABLE A-8: ZIGBEE PUBLIC PROFILE IDS AND CORRESPONDING APPLICATION DESCRIPTION

Descriptor Name	Status	Description
Node	Mandatory	Type and capabilities of the node.
Node power	Mandatory	Node power characteristics.
Simple	Mandatory	Device descriptions contained in node.
Complex	Optional	Further information about the device descriptions.
User	Optional	User-definable descriptor

Node Descriptor mainly describes the type and capabilities of a node. The following table shows the information conveyed in a node descriptor. The complex descriptor available field indicates a complex descriptor is available for further discovery of more information conveyed by complex descriptor.

TABLE A-9: FIELDS OF NODE DESCRIPTOR

Field Name	Length (Bits)
Logical type	3
Complex descriptor available	1
User descriptor available	1
Reserved	3
APS flags	3
Frequency band	5
MAC capability flags	8
Manufacturer code	16
Maximum buffer size	8
Maximum incoming transfer size	16
Server mask	16
Maximum outgoing transfer size	16
Descriptor capability field	8

Simple Descriptor describes the information listed in the following table. This descriptor associates an endpoint with the application which is described by Application profile identifier and Application device identifier. The application input cluster list and the application output cluster list are cooperatively used with profile identifier to realise service discovery.

TABLE A-10: FIELDS OF SIMPLE DESCRIPTOR

Field Name	Length (Bits)
Endpoint	8
Application profile identifier	16
Application device identifier	16
Application device version	4
Reserved	4
Application input cluster count	8
Application input cluster list	16*I (where *I* is the value of the application input cluster count)
Application output cluster count	8
Application output cluster list	16*o (where o is the value of the application output cluster count)

Complex Descriptor contains extended information for each of the device descriptions contained in this node. The use of the complex descriptor is optional; the availableness of a complex descriptor is indicated in a node descriptor.

TABLE A-11: FIELDS OF THE COMPLEX DESCRIPTOR

Field Name	XML Tag	Data Type
Language and character set	<languageChar>	
Manufacturer name	<manufacturerName>	Character string
Model name	<modelName>	Character string
Serial number	<serialNumber>	Character string
Device URL	<deviceURL>	Character string
Icon	<icon>	Octet string
Icon URL	<outliner>	Character string
Reserved	-	-

To enable service discovery, Zigbee Alliance has specified public profiles, and corresponding cluster IDs and Device IDs. The following table lists the public profile IDs with which an application is profiled.

TABLE A-12: ZIGBEE PUBLIC PROFILE IDS AND CORRESPONDING APPLICATION DESCRIPTION

Zigbee Public Profile IDs	Description
0x0101	Industrial plant monitoring
0x0104	Home Automation
0x105	Commercial Building Automation
0x107	Telecom Applications
0x108	Personal Home and Hospital Care
0x109	Advanced Metering Initiative

Each public profile associates with a specified list of clusters and devices. Taking the Home Automation profile as an example, the following two tables list selected devices and clusters related to home automation.

TABLE A-13: DEVICES SPECIFIED IN THE HOME AUTOMATION PROFILE

Device	Device ID
Range Extender	0x0008
Mains Power Outlet	0x0009
Door Lock	0x000A
Door Lock Controller	0x000B
Simple Sensor	0x000C
Consumption Awareness Device	0x000D
Home Gateway	0x0050
Smart plug	0x0051
White Goods	0x0052
Meter Interface	0x0053
Flow Sensor	0x0306
Mini Split AC	0x0307
Reserved	0x0404-0xFFFF

TABLE A-14: CLUSTERS USED IN THE HA PROFILE

Functional Domain	Cluster Name	Cluster ID
General	Basic	0x0000
General	Identify	0x0003
General	Groups	0x0004
General	Scenes	0x0005
General	On/Off Switch Configuration	0x0007
General	Power Profile	0x001a
General	EN50523Appliance Control	0x001b
General	Poll Control	0x0020
Closures	Shade Configuration	0x0100
Closures	Door Lock	0x0101
Closures	Window Covering	0x0102
HVAC	Pump Configuration and Control	0x0200
HVAC	Thermostat	0x0201
HVAC	Fan Control	0x0202
HVAC	Thermostat User Interface Configuration	0x0204
Lighting	Colour Control	0x0300
Measurement & Sensing	Illuminance Measurement	0x0400
Measurement & Sensing	Illuminance Level Sensing	0x0401
Measurement & Sensing	Temperature Measurement	0x0402
Measurement & Sensing	Pressure Measurement	0x0403
Measurement & Sensing	Flow Measurement	0x0404
Measurement & Sensing	Relative Humidity Measurement	0x0405
Measurement & Sensing	Occupancy Sensing	0x0406

Security and Safety	IAS Zone	0x0500
Security and Safety	IAS ACE	0x0501
Security and Safety	IAS WD	0x0502
Smart Energy	Metering	0x0702
Home Automation	EN50523 Appliance Identification	0x0b00
Home Automation	Meter Identification	0x0b01
Home Automation	EN50523 Appliance events and Alert	0x0b02
Home Automation	Appliance statistics	0x0b03
Home Automation	Electricity Measurement	0x0b04
Home Automation	Diagnostics	0x0b05

A Zigbee node with EFS resource needs to discover a service providing both Internet access and EFS and OCS access information. Taking the Home Automation scenario as an example, the Zigbee node can discover a Zigbee node providing Internet access by specifying the simple descriptor with profile ID 0x0104 (home automation) and with a device ID 0x0050 (home gateway). However, current Zigbee specifications lack the consideration of EFS and OCS design. Therefore, to enable discovery of EFS and OCS access information, we propose to add the two fields to Complex descriptor as shown in the following table. During service discovery, the node descriptor in a discovery response should set the complex descriptor available indicator so as to guide the requesting Zigbee node to further discover the complex descriptor.

TABLE A-15: PROPOSED EFS-ENABLED FIELDS OF THE COMPLEX DESCRIPTOR

Field Name	XML Tag	Data Type
EFS URL	<efsURL>	Character string
OCS Entry Point URL	<ocsEntryPointURL>	Character string

The method mentioned above merely provides a way to enable an EFS resource to access EFS and/or OCS. One also can find another profile together with a device ID which provides Internet access. Under such condition, as long as the EFS and OCS access information is configured in the corresponding complex descriptor, the EFS resource is still able to access EFS and/or OCS. The means to discover an EFS resource is not limited to what is described above One can still design other means to realise the EFS resource discovery, e.g., defining a new profile.

A.12 Bluetooth Service Discovery Protocol

Bluetooth Service Discovery Protocol (SDP) is another wireless low power protocol, but mainly used in mobile ad-hoc environments such as between a mobile phone and headsets or car stereo systems. Bluetooth device discovery is performed by periodically broadcasting and scanning for inquiries.

Once the device address is known, the connection is established through pairing and only then the information about services available on devices is provided. Even though the latest Bluetooth specification (BLE or version 4.0) achieves low energy consumption and high pairing speed, SDP only works for Bluetooth devices which make it unsuitable for other types of low power devices.

Discovering the EFS via Bluetooth [42] interfaces can be done via SDP protocol which was specified by the Bluetooth SIG in December 2016. SDP provides a means for SDP clients to discover the existence of services provided by SDP servers. To discover a service, an SDP client issues an SDP request to an SDP server to retrieve service information of the service. A service is described by a service record, which contains a list of attributes to characterise the service.

To describe a service in an organised and extensible way, the concept of service class is introduced. ServiceClassIDList attribute in a service record is a list of service class IDs, which enables service hierarchy organisation. The following example describes the usage and meaning of ServiceClassIDList attribute. A printer service in colour postscript with duplex capability might conform to 4 service class definitions so that the corresponding ServiceClassIDList includes four service classes as follows:

- DuplexColorPostscriptPrinterServiceClassID;

- ColorPostscriptPrinterServiceClassID;
- PostscriptPrinterServiceClassID;
- PrinterServiceClassID.

Such printer service inherits the attributes from PrinterServiceClass, PostscriptPrinterServiceClass and ColorPostscriptPrinterServiceClass, and defines attributes to describe the duplex capability in DuplexColorPostscriptPrinterServiceClass.

The following table selectively lists service classes defined in the Bluetooth specifications. Each of service classes is identified by its own UUID (Universally unique identifier), which plays a key role in service discovery.

TABLE A-16: SERVICE CLASS PROFILE IDENTIFIERS

Service Class Name	UUID	Specification
ServiceDiscoveryServerServiceClassID	0x1000	Bluetooth Core Specification
BrowseGroupDescriptorServiceClassID	0x1001	
SerialPort	0x1101	Serial Port Profile (SPP)
OBEXObjectPush	0x1105	Object Push Profile (OPP)
OBEXFileTransfer	0x1106	File Transfer Profile (FTP)
Headset	0x1108	Headset Profile (HSP)
AudioSource	0x110A	Advanced Audio Distribution Profile (A2DP)
AudioSink	0x110B	
A/V_RemoteControlTarget	0x110C	Audio/Video Remote Control Profile (AVRCP)
A/V_RemoteControl	0x110E	
Headset – Audio Gateway (AG)	0x1112	Headset Profile (HSP)
PANU	0x1115	Personal Area Networking Profile (PAN)
DirectPrinting	0x1118	Basic Printing Profile (BPP)
ImagingResponder	0x111B	
ImagingAutomaticArchive	0x111C	
ImagingReferencedObjects	0x111D	
Handsfree	0x111E	Hands-Free Profile (HFP)
HandsfreeAudioGateway	0x111F	
SIM_Access	0x112D	SIM Access Profile (SAP)
Phonebook Access – PCE	0x112E	Phonebook Access Profile (PBAP)
Phonebook Access – PSE	0x112F	
Headset – HS	0x1131	Headset Profile (HSP)
Message Access Server	0x1132	Message Access Profile (MAP)
Message Notification Server	0x1133	
3D Display	0x1137	3D Synchronisation Profile (3DSP)
3D Glasses	0x1138	
VideoSource	0x1303	Video Distribution Profile (VDP)
VideoSink	0x1304	
VideoDistribution	0x1305	
HDP Source	0x1401	Health Device Profile (HDP)
HDP Sink	0x1402	
Reserved (Max value 0xFFFF)		

To carry out effective and efficient service discovery, two main operations are specified: searching and browsing. When a user is able to identify some desired characteristics of a target service, the searching operation is suitable for service discovery. To search for services, service search pattern is used, which is a list of UUIDs. A service search pattern matches a service record if the set of UUIDs in the service search pattern is a subset of UUIDs in the service record. Once a search matches, SDP client receives a service record handle. This handle can be used to request the values of specific attributes of the service record.

When a user is not able to precisely determine the characteristics of a target service, the browsing operation is helpful. To browse for services, an SDP client creates a service search pattern with the root browse group UUID included only. A service is listed at the root browse group as long as its BrowseGroupList attribute includes the UUID of the root browse group. One can use BrowseGroupList attribute to organise services in a hierarchical way so as to realise the browsing for services. The following table illustrates a service browsing hierarchy example. The Entertainment, News, Reference Games and Movies are of BrowseGroupDescriptor service class with BrowseGroupList attribute set to PublicBrowseRoot. The root browsing shows the five services. If the user further browses the News service, the services with BrowseGroupList attribute set to NewsID shows in the browsing result.

TABLE A-17: SERVICE BROWSING HIERARCHY

Service Name	Service Class	Attribute Name	Attribute Value
Entertainment	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	EntertainmentID
News	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	NewsID
Reference	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	ReferenceID
Games	BrowseGroupDescriptor	BrowseGroupList	EntertainmentID
		GroupID	GamesID
Movies	BrowseGroupDescriptor	BrowseGroupList	EntertainmentID
		GroupID	MoviesID
Starcraft	Video Game Class ID	BrowseGroupList	GamesID
A Bug's Life	Movie Class ID	BrowseGroupList	MovieID
Dictionary Z	Dictionary Class ID	BrowseGroupList	ReferenceID
Encyclopedia X	Encyclopedia Class ID	BrowseGroupList	ReferenceID
New York Times	Newspaper ID	BrowseGroupList	NewsID
London Times	Newspaper ID	BrowseGroupList	NewsID

To reach an EFS service platform or OCS, an EFS resource node should discover a Bluetooth device with Internet access service (i.e., PAN service). In addition, the EFS resource node should obtain EFS URL for accessing EFS service platform and/or OCS entry point URL for OCS access. The personal area networking profile has been specified to support internet access with service class UUIDs: 0x1115 for PAN User (PANU) and PANU service, 0x1116 for Network Access Point (NAP) and NAP service and 0x1117 for Group Ad-hoc Network (GN) and GN service. The following table lists the attributes being used for PAN service.

TABLE A-18: APPLICABLE ATTRIBUTES IN PERSONAL AREA NETWORKING PROFILE

Attribute Name	Attribute ID
IpSubnet (Not used in PAN v1.0)	0x0200
SecurityDescription	0x030A
NetAccessType	0x030B
MaxNetAccessrate	0x030C
Ipv4Subnet	0x030D
Ipv6Subnet	0x030E

To provide access to EFS service platform and OCS, we propose to add two attributes the profile shown in the following table. These two attributes are applicable if a service provided by a Bluetooth device is able to provide internet access.

TABLE A-19: EFS-SPECIFIC ATTRIBUTE DEFINITIONS

Attribute Name	Attribute ID	Description
EFS URI	0x310	Provides a list of one or more URI for uniquely identifying the EFS available on the network

OCS Entry Point URL	0x311	Provides a URL for the Orchestration and Control System which can be used as entry point for integrating the resource into the existing EFS

To correctly identify an EFS enabled service via Service Searching, a discoverable attribute, say with attribute name to be EFS information service, should be specified with a value of a UUID. To discover an EFS enabled service via Service Browsing, a new service class is specified in the Service Browsing Hierarchy shown in the following table.

TABLE A-20: EXAMPLE OF SERVICE BROWSING HIERARCHY WITH EFS & OCS ACCESS SERVICE

Service Name	Service Class	Attribute Name	Attribute Value
Reference	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		Group ID	ReferenceID
EFS & OCS Access	ReferenceID	BrowseGroupList	ReferenceID
		EFS URL	EFS access URL
		OCS Entry Point URL	OCS access URL

The means to discover an EFS resource is not limited to what is described above One can still design other means to realise the EFS resource discovery, e.g., defining a new profile.

A.13 Preboot Execution Environment

The Preboot Execution Environment (PXE) uses client-server protocols such as DHCP and TFTP to boot operating systems and install software retrieving from the network on client machines. It is considered an effective way of managing desktop PCs across the network. Despite of its complexity, PXE reduces the cost of supporting physical equipment. The required components of PXE are as follows.

1. PXE-enabled NIC card. The firmware chip on the NIC must include the PXE code to be able to perform PXE. The NIC card must support wake-on-LAN signal. For using PXE it is needed to enable PXE and wake-on-LAN features in the PC's BIOS. Note that the retail PCs rarely have support for PXE and wake-on-LAN. Another issue is that the PXE does not support wireless NICs.
2. A server machine containing the OS boot image and the OS CD's (or the required software image CD's).
3. DHCP server from which to boot successfully.

DHCP protocol is exploited to provide appropriate client network parameters and specifically the IP address of the TFTP server hosting, ready for download, the initial bootstrap program (NBP) and complementary files.

PXE network booting cannot work on a subnet containing multiple DHCP servers.

A.14 Zeroconf

Zero Configuration Networking (zeroconf) is set of protocols and techniques for automatic network configuration in a local area network. It accomplishes network setting without manual configuration or special servers such as Domain Name System (DNS) and Dynamic Host Configuration Protocol (DHCP). Zeroconf can be implemented in several ways. However, the three main tasks comprising zeroconf are address auto configuration, name-to-address translation and service discovery.

For address auto configuration, zeroconf uses link-local addressing to replace DHCP. IPv4 reserves address block 169.254.0.0/16 for link-local address but an IPv4 host only uses it as a last resort when a DHCP server is unavailable. IPv6 uses the prefix fe80::/10 for link-local address and every host is required to configure a link-local address. Zeroconf enables a device to dynamically configure a link-local address. On top of that, determining the subnet mask to use, performing duplicate address detection and coping with address collision are also required for address auto configuration.

Zeroconf uses multicast domain name service (mDNS) for name-to-address translation. It allows devices to resolve name locally without a DNS server. A device can select a domain name in local namespace and then announce the name by sending a UDP multicast query message. Each device in this network maintains a list of DNS resource records, i.e., mDNS cache, which is updated when the device receives DNS query. Another implementation by Microsoft is Link-Local Multicast Name Resolution (LLMNR) protocol which is similar to mDNS protocol.

Finally, there are two primary mechanisms for service discovery. One is Service Location Protocol (SLP) (Sec. 1.1.4). The other is DNS-based service discovery (DNS-SD) protocol, which allows devices to look up services via DNS. There are three record types in DNS-SD: SRV record, PTR record, and TXT record. Service is described using DNS SRV and TXT records. A client can use DNS PTR record to specify the type of service, the transport protocol, and the domain name to discover the list of available service instances that match the request. Other protocols for service discovery also use multicasting extensively, such as Simple Service Discovery Protocol (SSDP) and Web Services Dynamic Discovery (WS-Discovery) protocol.

A.15 IoT COAP

The Constrained Application Protocol (CoAP) is a request/reply protocol for constrained devices and networks. These devices are often 8-bit microcontrollers with small memories, and the networks typically have high error rates and low throughput. CoAP nodes can act as clients or as servers, where the client issues requests to a server, and receives replies from the server. A node can fulfil both roles at the same time.

CoAP allows a client to discover resources available in a server by issuing a GET request for a well-known URI to the default CoAP port number (5683). The response is a list containing at least the minimal set of resources from which all resources can be discovered. Issuing a GET request for some non-leaf resources in the hierarchy allows (further) discovery of that part of the hierarchy. Resource discovery can be refined by specifying a query on resource attributes, reducing the size of the response. Metadata on the resource can be included the response or be linked to by specifying the URI at which additional information can be found.

A client can discover CoAP servers by multicasting the GET request. All multicast-capable servers with resources matching the query will then respond to the request. The multicast address is specified in RFC7252 and is 224.0.1.187 for IPv4 and ff0x::fd for IPv6 (the “x” stands for the multicast scope, either link-local and site-local).

CoAP support both proxying and caching, and explicitly also support proxying in combination with multicast. It is therefore possible to build a network of mostly-asleep nodes that rely on a few more powerful nodes for satisfying (discovery) requests most of the time.

Appendix B: Monitoring platforms and tools

In this section, a list of platforms and tools is shown. Tools are separated by categories, but many tools can be used for different environments and scenarios.

B.1 Bare Metal

NAGIOS

Nagios[7] is an open-source monitoring system which allow identify and resolve infrastructure problems before they affect critical processes. It provides a centralised view of entire IT infrastructure and detailed up-to-date status information. Nagios is able to:

Monitoring: Monitors system metrics, network protocols, applications, services, servers and network infrastructure.

- Alerting: generates alerts of failures once critical infrastructure components fail, they can be delivered via email, SMS or custom scripts, and detailed reports of outages.
- Self-healing: event handlers allow automatic restart of failed applications and services and its extendable architecture allows integration with third-party applications, with multiple APIs.
- Reporting: reports keep a record of historical outages, events, notifications, and alert response for later review. These reports help to ensure that SLA agreements are been met.

It needed almost a system with 1 GHz Processor, 1 GB RAM and 8 GB HDD, but is recommended a 2+ GHz Processor, 2 GB RAM, 40 GB HD and RAID 5 Drive Configuration. The table below provides hardware recommendations based on a node (host) to service ratio of 1:5.

TABLE B-1: HARDWARE RECOMMENDATIONS

Monitored Nodes/Hosts	Monitored Services	HD Space	CPU Cores	RAM
50	250	40GB	1-2	4GB
100	500	80GB	2-4	4-8GB
>500	>2500	120GB	>4	>8GB

PERFORMANCE CO-PILOT (PCP)

PCP[8] is a system performance and analysis framework which can collect multiple metrics from a variety of operating systems in real-time or by using historical data.

A PCP feature is data collecting. Each host being monitored requires a Performance Metrics Domain Agent (PMDA), which is responsible for collecting performance measurements from that domain. There are distinct types of agents:

- Kernel agents: imports a broad range of performance data from kernels. E.g. CPU, disk, memory, swapping, network, NFS, RPC, file systems, and per-process statistics.
- Other agents: Import performance data from: Databases, Log files, Web servers, Mail systems, Cisco routers, Search engines, Cluster infrastructure, Application instrumentation (including the JVM) and the PCP infrastructure itself

Another feature is data analysing. PCP uses a single API for accessing performance data hiding the source of the data. Integrated archive logging and replay use same API from real-time metrics and historical data from and archive. Furthermore, it allows automated monitoring through rule-based language and interpreter.

[7] https://www.nagios.org/
[8] http://pcp.io

PCP has a distributed model which allows leveraging client-server architecture, allows multiple clients to monitor the same host and a single client to monitor multiple hosts in a cluster.

ICINGA

Icinga[9] is an extensible monitoring system that checks the availability of your infrastructure resources with notifications for outages. It was originally created as a fork of Nagios, adding a GUI, additional database connectors and a REST API to allow integration of extensions without modification of the application core. Icinga allows monitor network services such as SMTP, POP3, HTTP and others, along with host resources including CPU load, disk usage and server components (from switches, routers and temperature, to humidity sensors).

Best Icinga feature is their easy configuration. It uses a rule-driven configuration format that is user-friendly and keeps configuration work to a minimum by defining templates to apply to configuration objects. Icinga offers three distinct command types; check, notification and event commands. Each can be given default values, custom attributes, runtime macros and conditional behaviours.

Incinga is a single system, but it can make stronger via plugins. The simple plug-in system allows easily develop service checks, with optional interfaces, reporting modules and template based reports.

OPENNMS

OpenNMS[10] is a carrier-grade, highly integrated and open source platform. It is capable of collect performance metrics from industry standard agents via SNMP, JMX, WMI, NRPE, NSClient++ and XMP. Gather performance data from applications via customizable generic collectors with HTTP, JDBC, XML or JSON.

Topology discovery is their most powerful feature. It is able to discover layer 2 network topologies based on SNMP information from industry standards like LLDP, CDP and Bridge-MIB discovery. Moreover, it supports, supports layer 3 routing topology discovery based on OSPF and IS-IS.

OpenNMS has an alarm management system which when an event occurs the alarm is raised, and a notification is sent across multiple groups of users until it is acknowledged and addressed. Furthermore, it has detecting service outages from the client perspective and reports back to the central OpenNMS application, this is used to ensure centrally-hosted applications are always available.

B.2 Virtual Machine Monitoring Tools

THE CEILOMETER (OPENSTACK)

This project[11] is a data collection service that provides the ability to normalise and transform data across all current OpenStack core components with work underway to support future OpenStack components. Ceilometer is a component of the Telemetry project. Its data can be used to provide customer billing, resource tracking, and alarming capabilities across all OpenStack core components.

GNOCCHI

It is a system[12] capable of storing the metrics data in an efficient manner. It is a spin-off project of Ceilometer designed to provide a TDBaaS (Time-series Database as a Service) suitable for the specific needs of the project.

[9] https://icinga.com
[10] https://opennms.org/en
[11] https://docs.openstack.org/ceilometer/latest/
[12] https://wiki.openstack.org/wiki/Gnocchi

SNAP

This is an open-source telemetry framework[13]. Telemetry is simply information about your data centre systems. It covers anything and everything you can collect, from basic descriptive information, to performance counters and statistics, to log file entries. Snap can collect metrics from Docker, virtual machines, bare metal, etc. and extract it to multiple systems.

B.3 Log Collector Systems

LOGSTASH

It is an open source[14], server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favourite "stash." Normally, is used with ElasticSearch and Kibana to analyse and visualisation the data.

FLUME

This is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving substantial amounts of log data[15]. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tuneable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.

FLUENTD

This is an open source data collector for unified logging layer[16]. Fluentd allows to unify data collection and consumption for a better use and understanding of data.

B.4 Data Analytics and Visualisation System

GRAFANA

It is an open platform for beautiful analytics and monitoring[17]. Essentially, Grafana helps users to easily create and edit dashboards to visualise the data from multiple data sources such as, graphite, influxdb, Prometheus, elasticsearch, etc.

B.5 Monitor Network Metrics

The traffic stats from a switch can be analysed with sFlow tool. sFlow[18] is a general-purpose network traffic measurement system technology. sFlow is designed to be embedded in any network device and to provide continuous statistics on any protocol (L2, L3, L4, and up to L7), so that all traffic throughout a network can be accurately characterised and monitored. These statistics are essential for congestion control, troubleshooting, security surveillance, network planning, etc.

[13] https://snap-telemetry.io/
[14] https://www.elastic.co/products/logstash
[15] https://flume.apache.org
[16] https://www.fluentd.org
[17] https://grafana.com
[18] https://sflow.org

Appendix C: Trusted and untrusted non-3GPP network accesses

The 3GPP standard introduces two classes of access, namely trusted and untrusted non-3GPP access, such as Wi-Fi, WiMax and fixed networks. For the sake of clarity, we first describe the mechanisms to enable Wi-Fi access in both the 3GPP trusted and untrusted scenario, and then we present the envisioned solution to allow a connected car to be integrated into the EFS through a non-3GPP RAT.

Trusted non-3GPP Wi-Fi access was introduced in 3GPP Rel. 8 and is often considered as an operator-built Wi-Fi access with encrypted Wi-Fi radio access network (RAN) and secure authentication method. To be considered trusted, the Wi-Fi access must fulfil the following requirements:

- 802.1x-based authentication which also requires RAN encryption
- 3GPP-based network access using EAP method for authentication
- IPv4 and/or IPv6

In this case, UE is connected via a Trusted Wireless Access Gateway (TWAG) in the Wi-Fi core, which is connected to the P-GW in the Evolved Packet Core (EPC) through a secure tunnel, such as GPRS Tunnelling Protocol (GTP), Mobile IP (MIP) or Proxy Mobile IP (PMIP). To set up the GTP tunnel, a number of parameters in the subscriber profile are needed, including the user's IMSI, thus trusted 3GPP Wi-Fi access cannot be established by devices without SIM cards.

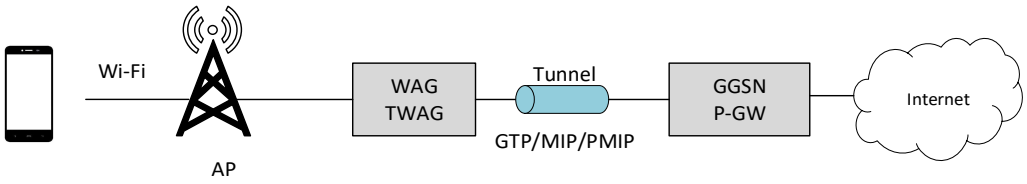

FIGURE C-1: 3GPP TRUSTED WI-FI ACCESS

By contrast, an untrusted non-3GPP Wi-Fi access is considered open and unsecured, and was first introduced in 3GPP Rel. 6, when Wi-Fi APs were characterised by limited security specifications. Typically, untrusted Wi-Fi accesses are employed for Wi-Fi Calling and comprise public hotspots, subscribers home and corporate Wi-Fi networks, as well as Wi-Fi access solutions not meeting sufficient security requirements, e.g., authentication and radio link encryption. Although this model does not require changes to the Wi-Fi RAN, it assumes that the UE is equipped with an IPSec client to establish a tunnel with the Evolved Packet Data Gateway (ePDG). The ePDG is then connected to the P-GW where each user session is transported through a secure tunnel (GTP or PMIP).

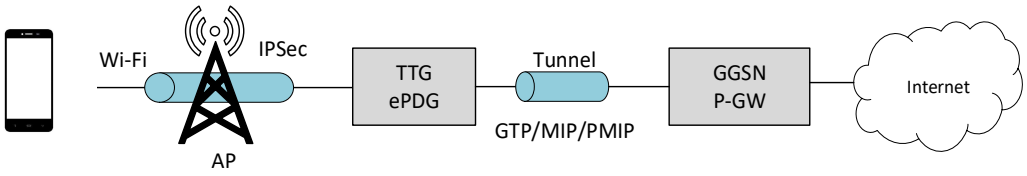

FIGURE C-2: 3GPP UNTRUSTED WI-FI ACCESS

C.2 Trusted and untrusted non-3GPP access in next-generation 5G systems

In next-generation 5G systems, Non-3GPP access networks will be connected to the 5G core network (nextGen Core) through the Non-3GPP Interworking Function (N3IWF). As shown in Figure C-3, this module interfaces to the nextGen Core network Control Plane (CP) and User Plane (UP) functions via NG2 and NG3, respectively. Furthermore, NWu represents the reference point between the UE and the N3IWF via untrusted non-3GPP access and is used for establishing an

IPSec connection between the two entities, and for carrying NAS signalling via the IPSec connection as well as supporting user-plane communication.

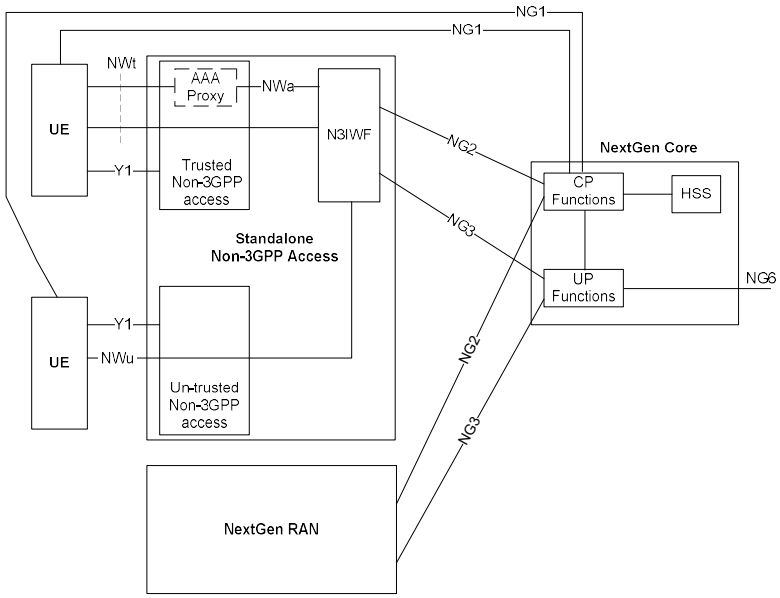

FIGURE C-3: ARCHITECTURE FOR ENABLING NON-3GPP ACCESS TO NEXTGEN CORE NETWORK