IoT-NGIN

D2.1
Enhancing IoT M2M/MCM Communications

WORKPACKAGE	WP2	**PROGRAMME IDENTIFIER**	H2020-ICT-2020-1
DOCUMENT	D2.1	**GRANT AGREEMENT ID**	957246
REVISION	V2.0	**START DATE OF THE PROJECT**	01/10/2020
DELIVERY DATE	30/11/2021 (revised version : 13/07/2022)	**DURATION**	3 YEARS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246

DISCLAIMER

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Pilroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelixis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP2
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	30/11/2021
ACTUAL DATE OF DELIVERY	21/12/2021 (revised version: 13/07/2022)
DOCUMENT TITLE	Enhancing IoT M2M/MCM Communications
AUTHOR(S)	SU, EDD, ABB, eBOS, CMC, I2CAT, RWTH
REVIEWER(S)	ATOS, SYN
ABSTRACT	SEE EXECUTIVE SUMMARY
HISTORY	SEE DOCUMENT HISTORY
KEYWORDS	5G, M2M, MCM, API

Document History

Version	Date	Contributor(s)	Description
V0.1	09/11/2021	RWTH, EDD	First draft
V0.2	17/11/2021	SU, EDD, ABB, eBOS, CMC, I2CAT, RWTH	Second draft and feedback from the partners
V0.3	15/11/2021	RWTH	Restructuring of chapter 4
V0.4	17/11/2021	EDD, RWTH	Updates in Chapter 4, Added section 1.2.3
V0.5	18/11/2021	EDD, RWTH, I2CAT	Chapter 4 Updates from discussion
V0.6	22/11/2021	SU, EDD, ABB, eBOS, CMC, I2CAT, RWTH	Third draft and feedback from the partners
V0.7	06/12/2021	SU	Final draft sent to the peer-reviewers
V0.8	10/12/2021	ATOS, SYN	Peer-reviewing
V0.9	15/12/2021	SU	Final revised draft sent to the partners
V.10	20/12/2021	SU, EDD, ABB, eBOS, CMC, I2CAT, RWTH	Partners' input and revisions integrated to the final draft
V1.0	21/12/2021	SU	Final version sent to the coordinator for submission
V2.0	13/07/2022	SU	Correction of the broken links

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

3GPP	3rd Generation Partnership Project
5G LAN	5G Local Area Network
5GS	5G system
AF	Application Function
AMBR	Allowed Maximum Bit Rate
API	Application Programming Interface
APN	Access Point Name
C-IoT	Cellular-IoT
COAP	Constrained Application Protocol
D2D	Device-to-Device
EC-GSM-IoT	Extended Coverage Global System for Mobile communications in the context of IoT
eMBB	Enhanced Mobile Broadband
eMTC	Enhanced Machine Type Communication
ETSI	European Telecommunications Standards Institute
FOTA	Firmware-Over-The-Air
GPRS	General Packet Radio Service
GUO	Graphical User Interface
IIOT	Industrial Internet of Things
IoT	Internet of Things
LAN	Local Area Network
LCM	Life Cycle Management
LoRa	Long Range
LPWA	Low Power Wide Area
LTE	Long-Term Evolution
LTE-M	Long-Term Evolution - Machine Type Communication
LwM2M	Lightweight Machine-to-Machine
M2M	Machine-to-Machine
MANO	Management and Orchestration
MCL	Maximum Coupling Loss
MCM	Machine Cloud Machine
mMTC	Massive Machine-Type Communication
MQTT	Message Queue Telemetry Transport
MTO	Multi-Tier Orchestrator
NB-IoT	Narrowband Internet of things
NEF	Network Exposure Function
NIDD	Non-IP Data Delivery
NR IIOT	NR Industrial Internet of Things
NR-RedCap	NR Reduced Capability

NSMS	Network Slice Management System
NTN	Non-Terrestrial Networks
OSM	OpenSource MANO
PAN	Personal Area Network
PLMN	Public Land mobile network
QCI	QoS Class Identifier
QoS	Quality of Service
RAN	Radio Acces Network
RAN-NSSMS	RAN-Network Subnet Slice Management System
REST	Representational State Transfer
RTE	Runtime Execution Environment
SCEF	Service Capability Exposure Function
SEAL	Service Enabler Architecture Layer
SM	Slice Manager
SNR	Signal-to-Noise Ratio
SOE	Slicing & Orchestration Engine
TCP	Transmission Control Protocol
TMF	Tele Management Forum
TSN	Time Sensitive Networking
UDP	User Datagram Protocol
UE	User Equipment
URLLC	Ultra-Reliable and Low Latency Communication
VIM	Virtual Infrastructure Manager
VM	Vitual Machine
WP	Work Package

Executive Summary

IoT-NGIN is exploring new solutions for enabling advanced services as part of the European next-generation Internet. Those services will stress different communication capabilities that are considered in order to provide a proof-of-concept of their relevance and efficiency. We start by assessing the evolution of the demand identified from the living labs perspectives. We can observe that they expose various requirements regarding the communication services on top of which they will be executed – for instance, coverage extension, 5G-assisted location, real-time services, edge-processing, or secured execution.

Therefore, we study enhanced IoT underlying technology offering IoT/5G optimization and a by-design secure edge cloud execution environment to support micro-services, offloading, and increase capability.

In this framework, the project has laid focus on several activities to address the expectations mentioned above:

- Relay strategy towards extended 5G coverage;
- Transforming 5G into TSN (Time-Sensitive Networking) end-to-end system;
- API as a communication substrate for IoT-NGIN services;
- Secure offloading of IoT Tasks.

Although they are developed in parallel, these activities are tightly related. This Deliverable provides the current progress achieved for all activities except the one dealing with secure offloading that will be presented in the next deliverable.

The progress of all activities is presented in this document. The next step will deliver a first implementation of the solutions described here. After that, we will use them to support in-lab testing supporting the IoT-NGIN methodology and provide evidence about the relevance and efficiency of the proposed solutions.

1 Introduction

1.1 Context and objectives

IoT-NGIN aims at leveraging existing technology strengths to develop advanced communication services, which will rely on the underlying digital infrastructure. This deliverable presents the objectives and progresses achieved by the activities carried out to support the demand illustrated by the use-cases. Different and complementary targets are explored, dealing with coverage extensions using device-to-device communications, time-sensitive networking, optimized 5G resource management, and secure offloading in edge and cloud computing.

This deliverable describes the progress of the activities related to the first three tasks that started roughly a year ago. It will be extended over time in order to provide a timely description of the solutions developed and evaluated.

The general objectives are to achieve:

- easy end-to-end access to infrastructure for services;
- vendor-independent service creation;
- lower barrier for service development in edge clouds;
- reduced time-to-market for service applications;
- the support of decentralization;
- security and scalability at the communication level.

The specific objectives are:

- relay strategy towards extended 5G coverage;
- transforming 5G into TSN (Time Sensitive Networking) end-to-end system;
- API as a communication substrate for IoT-NGIN services;
- secure offloading of IoT Tasks.

These activities are articulated with the rest of the project as presented in Figure 1. A first introduction of the challenges and methodology is presented in this chapter.

1.2 Challenges

In this section, we motivate each of the challenges we explore in the rest of the document. At the time of the writing of this deliverable, we have been working on four challenges that are central to achieve the objectives of the work package.

1.2.1 Coverage extension through D2D

Efficiency is an omnipresent objective in IoT-NGIN. One of the challenges concerning the communication substrate is to provide connectivity to as many nodes as possible. In a real-world setup, such nodes may undergo one of the following situations: (i) fall outside the coverage zone of a 5G cellular antenna, (ii) observe high interference due to surrounding communications, and (iii) experience a poorer communication experience when compared to neighbouring nodes. In such a context, a promising strategy is to rely on

device-to-device communications to extend the cellular infrastructure so that nodes can benefit from other channels. At the same time, the network can achieve improved overall performance.

In the IoT-NGIN project, we address this problem by proposing a measurement-based approach to determine whether nodes dispose of device-to-device communication capabilities and, if so, whether they are efficient enough to perform an expected task. As detailed in Section 2, we focus on the measurement aspect. The other elements of the problem will be subject to future deliverables.

1.2.2 Optimizations of next-generation MCM communications

Multiple verticals such as manufacturing, agriculture, smart cities, harbours, smart grids need robust and ubiquitous connectivity to accommodate for Industrial IoT (IIoT) requirements part of Industry 4.0 and other applications where low latency, high throughput, maximum service availability and reliability are essential. The adoption of Machine Cloud Machine (MCM) communications is mainly for increasing productivity and reducing downtime with automation. One of their main requirements for adopting the IIoT is having reliable data networks (100% availability) with sufficient capacity. However, manufacturing companies are currently facing problems because most public networks are not designed to face the challenges of an expanding IIoT. Public networks are designed and configured for consumer data and exhibit low resource scalability and inability to efficiently react to industrial service requirements stemming from heterogeneous communication applications. The industrial requirements entail different needs of latency, bandwidth/data rate, time synchronicity, reliability and security.

Current public mobile networks have not been specifically configured to overcome MCM with unpredictable traffic increase. Instead, the mobile networks are dimensioned based on peaks for busy hours. Thus, MCM requires different scalability and managing traffic shaping problems imposed by Industrial IoT (IIoT) and Industry 4.0 communication requirements. To address MCM communications, it is required to deploy private Narrow Band Internet of Things (NB-IOT) through network slices in order to divide a network connection into various virtual ones to provide resources to different network traffic types and support IIoT requirements for traffic shaping.

MCM must communicate with each other as Virtual Network thus 5G Local Area Network (5GLAN) and Time Sensitive Networking (TSN) functionality is required for delivering a 5G network designed for industrial usage.

1.2.3 Dynamic management of 5G resources

In the fifth generation of mobile networks, infrastructure management is more and more utilizing APIs between containerized functions and also more and more capabilities are exposed to end users. This provides high flexibility for the configuration of the network and the integration of new technologies. The new field of network function virtualization combined with high availability and massive machine type communication enable a multitude of new use cases. Especially the growing market of IoT Applications is a very promising domain. Depending on the specific implementation of 5G networks the APIs needed to expose certain functionalities can differ.

Therefore, the goal is to create an easy-to-use API that leverages the use of 5G features for IoT application developers and thus decrease time-to-market. The API will not be limited to a specific implementation of a 5G or Edge-Cloud environment and, thus, will increase the flexibility of the applications, utilizing it, preparing the IoT-NGIN landscape for the next level in the evolution of the demand. The goal is to encourage more IoT Application developers and verticals to utilize the emerging 5G technologies.

1.2.4 Secure framework for micro-services

Today, many cloud and edge environments are based on micro-services, which are managed by orchestration tools like Kubernetes [1]. These micro-services are typically provided as containers or with KubeVirt [2] as virtual machines (VM). A VM represents a self-contained computer, booting a standard operating system and running unmodified applications just like on a common physical machine. This layering of operating systems increases the overhead and thus limits the scalability. In contrast to VMs, containers share the host operating system and instead isolate the network, the file system, and the process group from each other. This decreases the overhead in comparison to VMs, but reduces also the isolation from the host operating systems and implicitly increase the security risk. A lightweight solution with a similar security behaviour should provide a runtime environment for Kubernetes, which runs classical containers and common virtual machines. Depending on demands and security requirements, the provider of micro-services is able to decide, which runtime is the best runtime for their use-cases.

To reduce the overhead of VMs, a library operating (libos) can be used, which links an operating system to the application and realizes a highly-optimized, single-address-space machine image. Single-address-space implies that only one process can be handled by the libos, but this is often enough to realize a distributed system consisting of a set of VMs with single micro-services.

The secure framework has to be compatible with existing technologies, simplifying the deployment of micro-services. In addition, it has to offer strong isolation based on virtual machines and reduce the overhead by using an appropriate flavour of libos. The IoT-NGIN-proposed solution will provide support for RustyHermit [3] in a "virtualized container" approach for Kubernetes. By supporting the Open Container Initiative (OCI) Runtime Specification [4] it will be guaranteed that the secure framework will be supported by existing orchestration tools.

1.3 Connections within the project

This work included in this deliverable took as input the user requirements, both functional and technical, obtained from WP1 and were translated by technology experts into technological specifications for each individual component. The output of this and future versions of this deliverable will feed into the component integration WP (WP6), the secure edge cloud execution will also feed into WP3 and WP5 where it will be united with the provision of Machine Learning as a service and the Digital Twins, respectively.

All these technologies will be integrated together in WP6 and tested to ensure compatibility and proper operation. Eventually, they will be fed into the Living Labs of WP7 for real-life testing. However, a feedback loop between WP6 and WP7 will allow space for problem-

solving on component integration issues and even investigate which components are ready to be implemented or not.

Finally, once the technologies will be implemented and tested in the Living Labs, those experiments will feed back to WP1 in order to evaluate the performance of the IoT-NGIN solution through the Benchmarking Verification Framework, in order to assess whether the user requirements have been met. Again, this will be a continuous process of multiple design and test cycles until the IoT-NGIN solution meets the user requirements evaluated by both Quality of Service and Quality of Experience metrics.

A visual map of the connections of WP2 with the rest of the WPs and the IoT-NGIN, in general, is shown in Figure 1below.

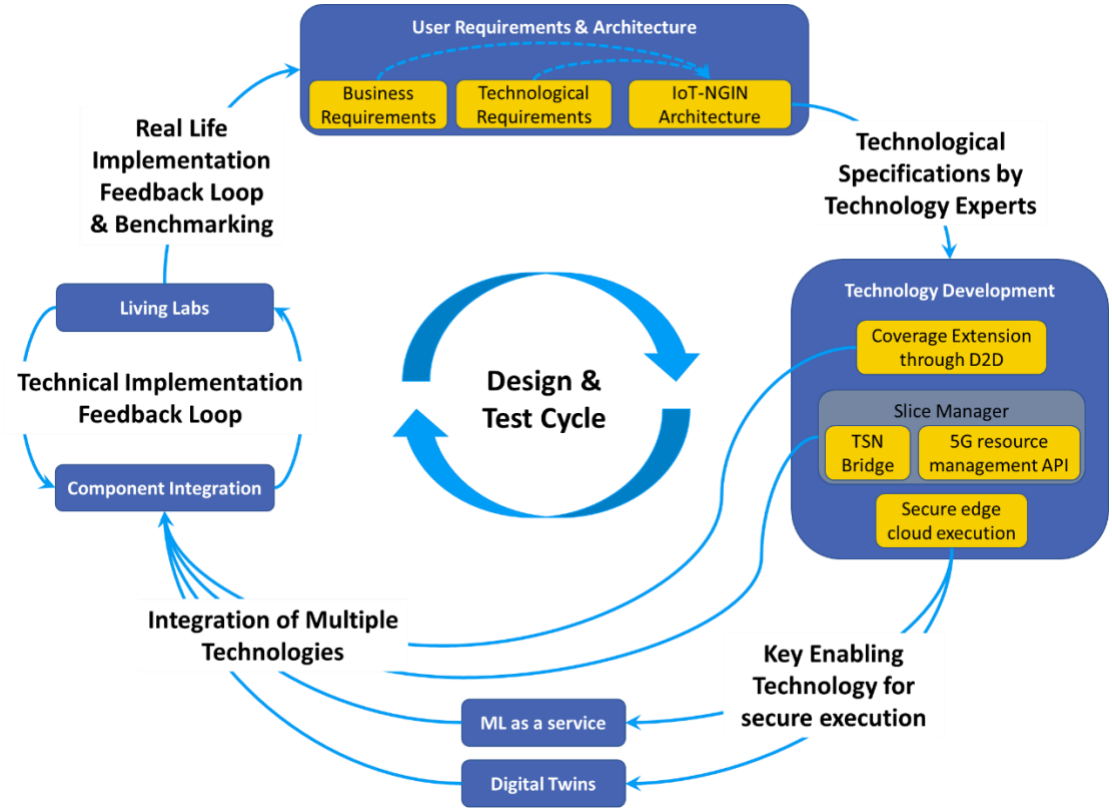

Figure 1: Visual map of the WP2 interdependencies within the IoT-NGIN project

1.4 Structure of the document

This deliverable is structured around the challenges that we have addressed during the first 14 months of the project. In Section 2, we present our contributions with regard to the coverage extension problem. We describe our work on machine-cloud-machine communications in Section 3. Our third original contribution, presented in Section 4, is the 5G API, a central component of IoT-NGIN. An important part of this deliverable is Section 5, where we present, besides some conclusion, the next steps of the project in WP2.

2 Coverage extension through D2D

One of the main objectives of this specific contribution is to explore optimal relay strategies. It further aims to analyse strategies to optimally deploy single/multi-hop relays in order to reduce the number of IoT devices that require large coverage enhancements and to ensure deeper coverage in 5G massive Machine Type Communications (mMTC), reducing the network overhead and covering practical market needs with reasonable cost. In the IoT-NGIN project, we are interested in one-hop coverage extension.

2.1 Background

5G offers two key service types, the Ultra-Reliable Low-Latency Communications (URLLC) and the massive Machine Type Communications (mMTC). URLLC, as its name states, serves critical communications requiring ultra-low latencies and impressive reliabilities with the trade-off on the scalability i.e., the number of devices it can serve simultaneously. mMTC, on the other hand, is at the end of the spectrum, offering increased scalability with connection densities of up to 10^6 devices/km^2 at the expense of decreased speeds and increased latencies [5]. As shown in Figure 2, it is envisaged that these two services will agglomerate into critical mMTC or scalable URLLC for beyond 5G with emerging use cases [6].

In most applications of mMTC, the requirements are usually related to the high number of connected devices, which are deployed in a wide area, sometimes in dense networks, and require sporadic communications without any latency or reliability needs. Some common indicators for such networks are [7]:

- Device densities of up to 10^6 devices/km^2 in urban environments.
- Maximum Coupling Loss (MCL) of 164 dB for wide coverage.
- Battery lifetimes of more than 10 years with energy capacities of 5 Wh.

There are several ways to achieve such massive connectivity, e.g., introducing subcarriers or tones for multiple access, however this aspect is not of interest in this work. Further, to lengthen the battery life of devices there are also multiple approaches such as wake up signals (WUS) in order to allow devices to avoid regularly paging checking and only start the procedure once WUS is received.

IoT-NGIN focuses on the coverage extension and provides two main approaches to achieve its goal. First, in order to extend the coverage range in an open environment and compensate for the penetration losses in a challenging indoor space with highly reliable communication, the network targets up to 164 dB MCL, which is 20 dB coverage enhancement compared to GSM and General Packet Radio Service (GPRS) [8]. To achieve this goal, except from operating in narrow bandwidth, the approach of transmission repetitions can be used by which the received Signal-to-Noise Ratio (SNR) can be enhanced so that data could be decoded even when the signal power is much lower than the noise power. An alternative approach to extend the coverage range is the single/multi hopping towards Device to Device (D2D) communications. This method can decrease the infrastructure costs, extend the coverage and maintain the high energy efficiency requirement of mMTC.

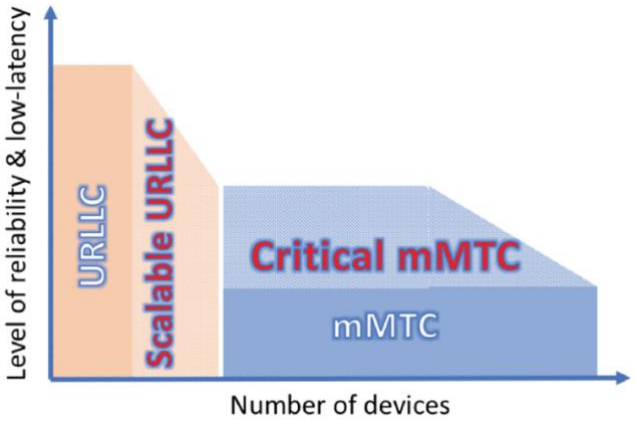

Figure 2: Reliability, latency & number of devices in a 5G network [9]

2.1.1 D2D Types

D2D communications is a technique that allows a device to communicate directly with another device, without or partially going through the network infrastructure. In simple terms, D2D communications provide the connection between the two wireless devices either directly, when both devices are in line of sight (classical ad-hoc network), or by employing multi-hop routing techniques when there is a blockage between the devices (non-line of sight). When the devices are communicating with each other, data transmission can be shared among the devices in the network, thus tremendously mitigating the traffic on the overall core network at the expense of lower bandwidth since only one device is really connected to the network. In applications such as NB IoT, where data rates or latencies are not critical, this approach can be very useful to connect more devices to the network whilst allowing a more optimised resource utilisation of the network [10]. Additionally, in a state of emergency, where the network infrastructure is down, the devices can still communicate with each other. A general scenario of a D2D communication over a IoT 5G network is illustrated in Figure 3.

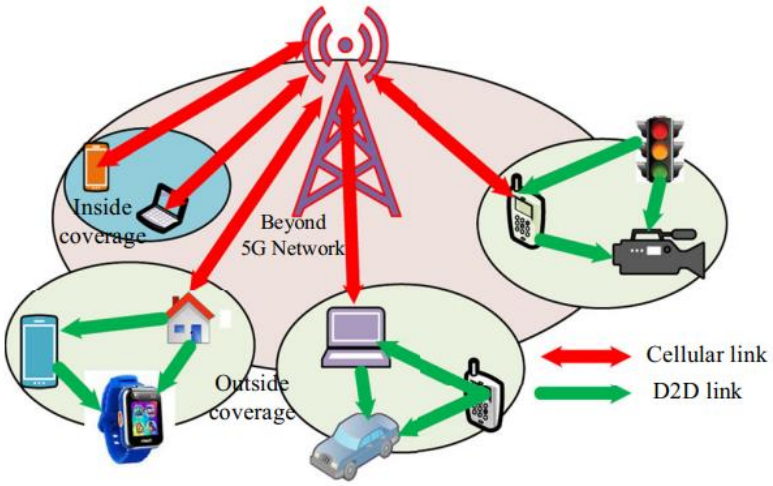

Figure 3: Network illustration with possible coverage expansion using D2D [11]

One of the most crucial aspects of the performance of D2D communications is routing. Proper routing will ensure that data is efficiently sent to the destination device (end-user). Routing mitigates the data packet's loss in a network by adjusting the path or route in the

response of dynamic network topology. Routing can be exemplified in a situation when the data is trying to reach a destination from one place (source) to another place (destination). Routing can be either direct or using a multi-hop path to the destination's device, depending on the current network topological information as can be seen in Figure 4. In a classical cellular network architecture, all the users are directly connected with a one-hop link to the network infrastructure. In contrast, for D2D communications, an intermediate device is needed due to the multi-hop link between source and destination device for data transmission.

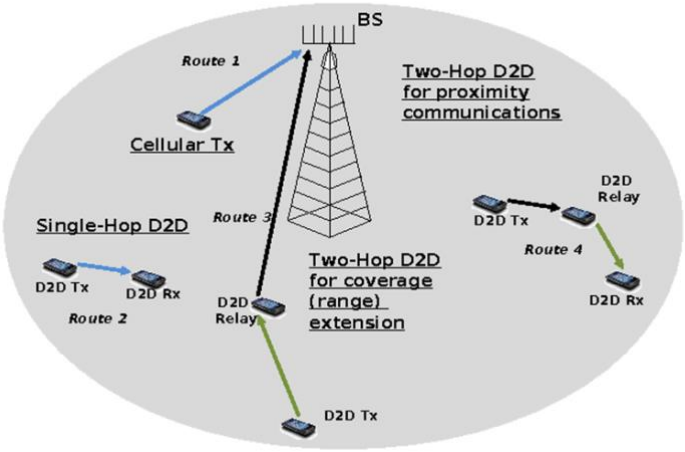

Figure 4: Routing types of D2D [12]

For this, a routing approach plays a significant role in efficient and reliable data transmission to the end-users. Due to the dynamic nature of the devices in the D2D communications, routing is pertinent to ensure standard QoS network performance is delivered. Based on the current state of network information, new routes can be decided on each time stats and this specifically, is one of the main research aspects of this work. Moreover, in D2D communications, energy constraints of the device are a big hurdle that should be appropriately addressed.

This dynamic movement of each and possible all the devices in the D2D communications is causing network instability. New network paths need to be re-established every time a device is moving out of range. This chain of events where the device keeps exiting and entering to the network, causes the network instability and deterioration of QoS performance [13]. In D2D communications, where the device is the key player, the queue length of data at a particular device will lead to traffic congestion. Longer waiting time at a specific device result in data transmission delays and increases traffic congestion in the network. At the same time, link failure occurs when the intermediate device moved [14] The source device needs to reselect and re-establish a new route to send the data packets based on the updated network topology. The selection of an optimum path from the source to destination devices in the D2D communication over a 5G cellular networks is limited due to the network instability, data packets traffic congestion, energy resource constraints, and link quality of the devices [15].

One of the limitations of the D2D communication is the data packets traffic congestion issue that takes place when there is an excessive amount of data packets injected on a single or particular device [16]. Hence, the routing schemes suffer from the load balancing amongst the relay devices, resulting in the deterioration of the overall network performance [17]. Moreover, energy resources and link quality of the device play an important role in providing a seamless connection with the end-users. In this scenario,

mobile devices are equipped with limited energy resources for their vital operation of data transmission. Thus, the connectivity of the devices suffers from packet drops as soon as the device's energy gets exhausted. Energy resources scarcity and limited processing power capacity of devices are probably the most important constraints in the development of an optimum path, which adversely impacts on network lifetime [18]. Similarly, the mobile devices in the network change their position in an unpredictable manner, with the probability of link failure in the established path. Due to the link failure, the data packet needs to undergo re-transmission with a new link, and this acquires more bandwidth in the network. Therefore, there is a need for optimal route selection in the D2D communication over IoT 5G cellular networks, which considers the above challenges to improve the overall network performance [19].

2.2 Methodology

We propose a simple, yet efficient, methodology to achieve coverage extension through D2D communications. For the network to decide which node to pick as a relay to reach another node that is outside the cellular coverage zone, it must obtain of the necessary information to decide which nodes could serve as relays. Such a selection process involves essentially three steps, as illustrated in Figure 5:

1. Characterization of the links between potential relays and destination nodes;
2. Assessment of the dynamics of the network;
3. Algorithm to select most appropriate relay, if any.

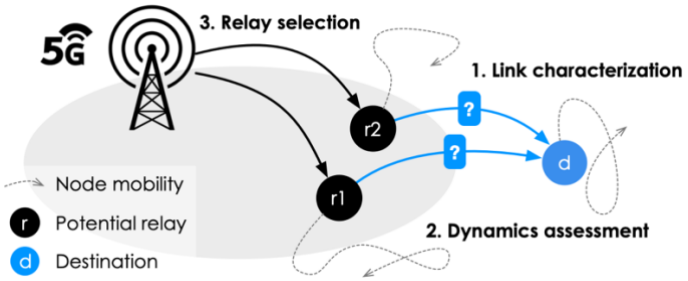

Figure 5: Steps toward relay selection for coverage extension in IoT-NGIN

During the first year of the project, we have focused on the first step of the process. We have decided to adopt an experimental approach instead of relying on models. We consider a scenario based on Android smartphones. Although the development is dedicated to one specific platform, the methodology is general and can be extended to other types of nodes. Because the Android operation system does not provide explicit measurements over D2D links, we had to develop a measurement tool whose goal is to perform some actions over a link and come up with characteristics such as throughput, latency, energy consumption, and stability, to cite a few. We call this tool AtomD and introduce it in Section 2.3.

2.3 AtomD

To analyse the actual scopes used in D2D, we have designed and implemented an Android application called AtomD, which starts a foreground D2D service using the Google

Nearby Connection API [20]. This API works at the application layer, so its scopes are not focused on transmitting individual transport layer packets, such as UDP and TCP, but on the transport of information fragments called chunks. It should be noted that these chunks can carry a maximum payload capacity of 2^{16} bytes or, equivalently, 64kB. AtomD performs a number of transmissions between two neighbouring nodes with the goal of characterizing the link between them.

AtomD allows two types of transmission: (i) a single chunk that carries the useful information to be transmitted (and to be processed immediately after its reception) and (ii) a block of data larger than 2^{16} bytes threshold, by subdividing the payload into multiple chunks. Note that the Google Nearby Connection API decides how many chunks will be used to transport the payload.

Once the chunks are prepared, they are queued in a transmission stack, identified with an id, and transmitted one by one to the receiver. The receiver decodes the target payload when all the expected chunks have been received.

In Figure 6, we present two screenshots of the AtomD application.

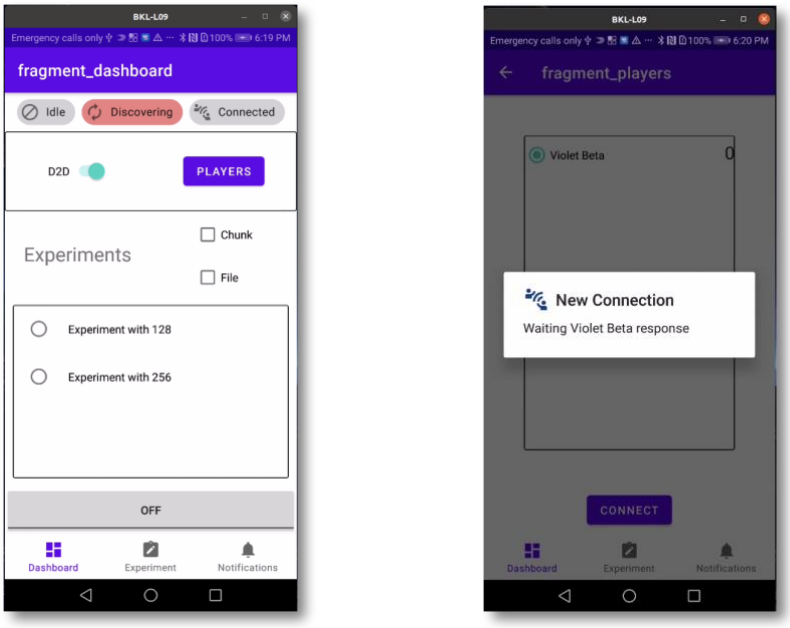

Figure 6: AtomD application

The figure on the left shows the phase where a node discovers neighbours while the figure on the right shows that a new connection is about to be established

AtomD is operational but still under development as some add-ons will make the application provide more features. For example, we plan to include a component to measure the connection delay before two neighbouring nodes can communicate. Also, we will include automatization functions to help users conduct experiments in an easier fashion. The first official release of the software is expected within the first quarter of 2022.

2.4 Experimental setup

The characterization of a link comes from the statistics of multiple transmissions between two nodes. In the remainder of this section, we report on the first set of experiments that we performed.

We established a connection between two devices and perform ten transmissions of a 250 MB-long binary file at multiple distances. Furthermore, we configured the devices to perform point-to-point connections to access the highest bandwidth.

The environment selected for these retransmissions is located in the periphery to avoid as much interference as possible. More specifically, at the coordinates `48°10'49.88''N, 1°32'14.396''W`. At this point, we established six points of distance as per Table 1.

Table 1: Positions of the nodes in the experimental setup

Distance (m)	N	W
0	48°10’49.88.88’’	1°32’ 14.396’’
20	48°10'49.351''	1°32' 13.841''
40	48°10'48.811''	1°32' 13.271''
60	48°10'48.266''	1°32' 12.737''
80	48°10'47.722''	1°32' 12.211''
100	48°10'47.207''	1°32' 11.641''

To introduce diversity in the system, we work with five Android devices, which are:

- OnePlus 5t [21];
- Honor View 10 [22];
- Samsung S20 [23];
- Samsung S8 [24];
- Xiaomi Redmi 9T [25].

2.5 Preliminary results

2.5.1 Throughput

A core part of the functionalities of AtomD is to record the receiving instance of a chunk in a database. This record contains the corresponding size of the chunk in bytes and its reception time in nanoseconds. In addition, we group each registered chunk by its corresponding payload ID. With this information, we obtain the number and size of chunks that each payload sent, plus the latency difference between each chunk.

We observe an example of this in Table 2, which shows the number of chunks required with their different sizes to send a 250 MB payload.

Table 2: Transmissions between different types of devices

Transmitter	Receiver	Chunk size (bytes)	# of chunks	Total # of bytes
Honor view 10	Samsung S20	1980	536	1061280
Honor view 10	Samsung S20	34580	1	34580
Honor view 10	Samsung S20	44108	1	44108
Honor view 10	Samsung S20	45696	58	2650368
Honor view 10	Samsung S20	65536	228824	14996209664
Xiaomi Redmi 9T	Samsung S20	1980	1560	3088800
Xiaomi Redmi 9T	Samsung S20	20348	1	20348
Xiaomi Redmi 9T	Samsung S20	45696	58	2650368
Xiaomi Redmi 9T	Samsung S20	62436	1	62436
Xiaomi Redmi 9T	Samsung S20	65536	228793	14994178048
One plus 5t	Samsung S20	1980	85	168300
One plus 5t	Samsung S20	10056	1	10056
One plus 5t	Samsung S20	44108	1	44108
One plus 5t	Samsung S20	45696	58	2650368
One plus 5t	Samsung S20	65536	228838	14997127168
Samsung S8	Samsung S20	1980	3	5940
Samsung S8	Samsung S20	3960	1	3960
Samsung S8	Samsung S20	35796	1	35796
Samsung S8	Samsung S20	45696	58	2650368
Samsung S8	Samsung S20	65536	228824	14996209664

To calculate the throughput T, we used the following equation:

$$T = \frac{8 \times avg(B) \times N}{2^{20}}$$

where $avg(B)$ is the average number of bytes sent per chunk and N is the number of times a chunk is sent in one second. Note that the result in bps is divided by 2^{20} (ISO IEC 80000 standard) to obtain Mbps [26].

2.5.2 Analysis

As mentioned above, we used five devices to send 250 MB in a point-to-point manner, performing ten retransmissions at different distances of up to 100 meters. As a result, we

obtained the following throughput results from the chunks recorded during these transmissions.

Throughput received by HONOR VIEW 10

In Figure 7, our target receiver is the Honor View 10. From here, we can observe that the minimum throughput approaches 25 Mbps, while most of the results range from 75 Mbps to 25Mbps. Furthermore, the Xiaomi Redmi 9T and Samsung S8 maintain a constant throughput beyond 40 meters, while the One Plus 5T and Samsung S20 are prone to sharp throughput rises. Finally, the Samsung S20 displays a better performance, maintaining its values above 50 Mbps, except for the receptions performed at 60 meters.

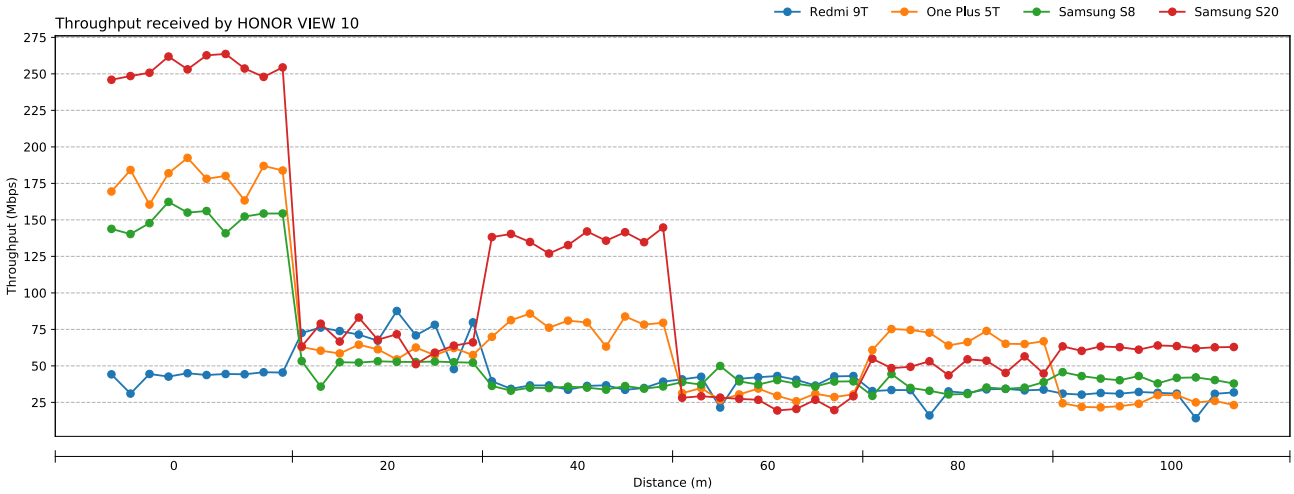

Figure 7: Throughput when the Honor View 10 is the receiver

Throughput received by ONE PLUS 5T

In the case of the One Plus 5T (Figure 8), we can see that the minimum throughput is close to 25 Mbps, as in the case of the Honor View 10. However, the throughput performance improves compared to this one, reaching average values that mostly pass 50 Mbps. On the other hand, we can observe a drop in performance during receptions located at 60m, possibly due to signal effects that involve the wireless's physical properties. Finally, the Samsung S8 and S20 have the best performance beyond 20 meters.

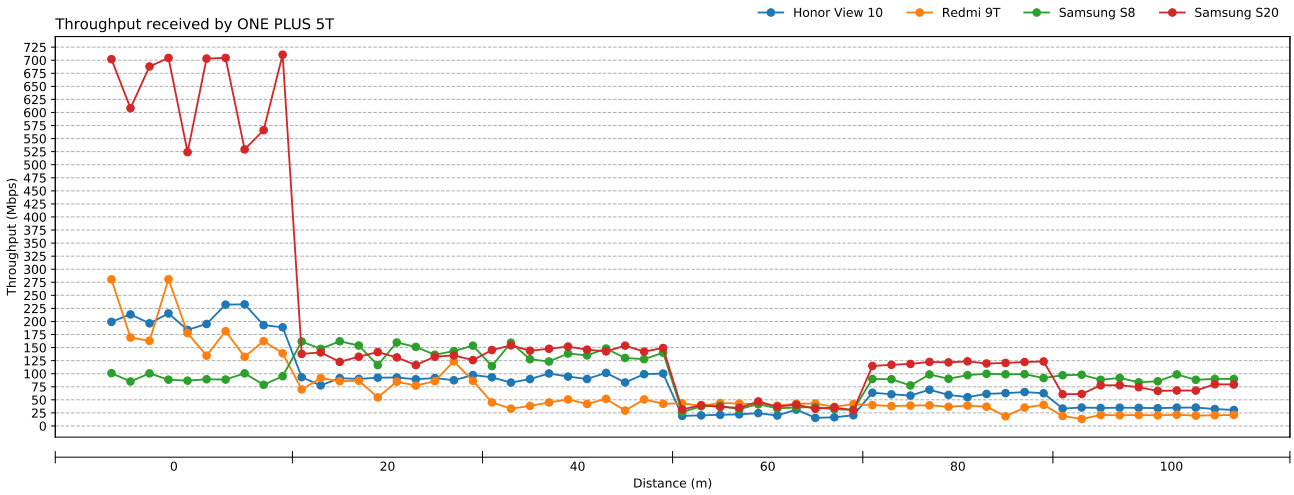

Figure 8: Throughput when the One Plus 5T is the receiver

Throughput received by REDMI 9T

In Figure 9, we find the Xiaomi Redmi 9T receiver, where we can see that the throughput obtained remains close to 25 Mbps.

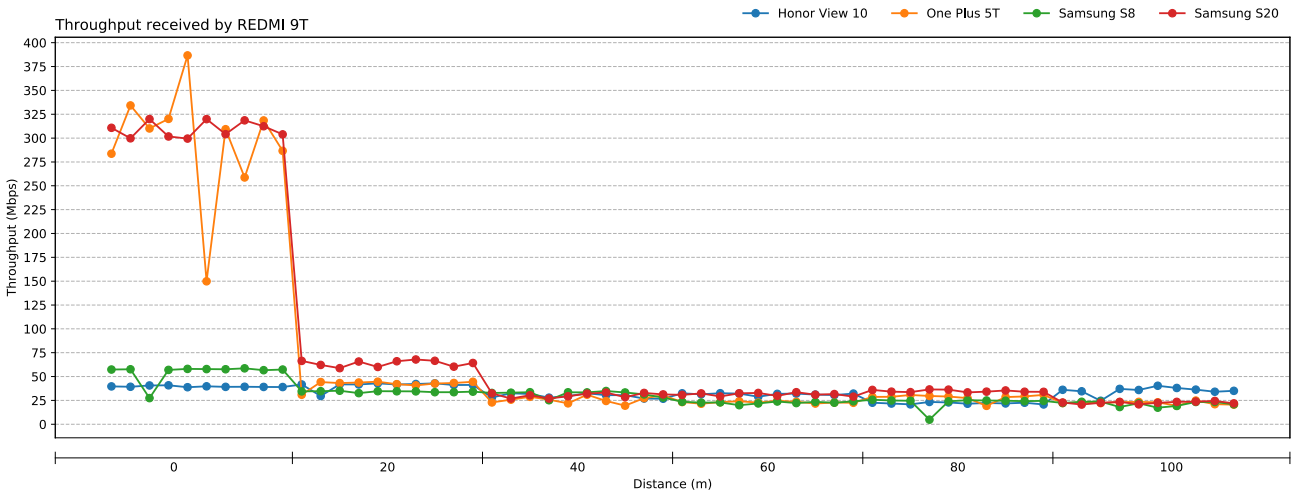

Figure 9: Throughput when the Redmi 9T is the receiver

Throughput received by SAMSUNG S8

In the case of the Samsung S8 (Figure 10), we can observe a linear decrease of the Samsung S20 as they move away. However, it should be noted that this decrease occurs while maintaining a throughput mostly above 75 Mbps. On the other hand, we have the Honor View 10 and the Xiaomi Redmi 9T, which maintain a similar behaviour from 20 meters. Finally, the One Plus 5T presents the highest values concerning throughput, reaching values ranging between 200 Mbps and 250 Mbps at 40 meters and values of 125 Mbps and 150 Mbps at 100 meters.

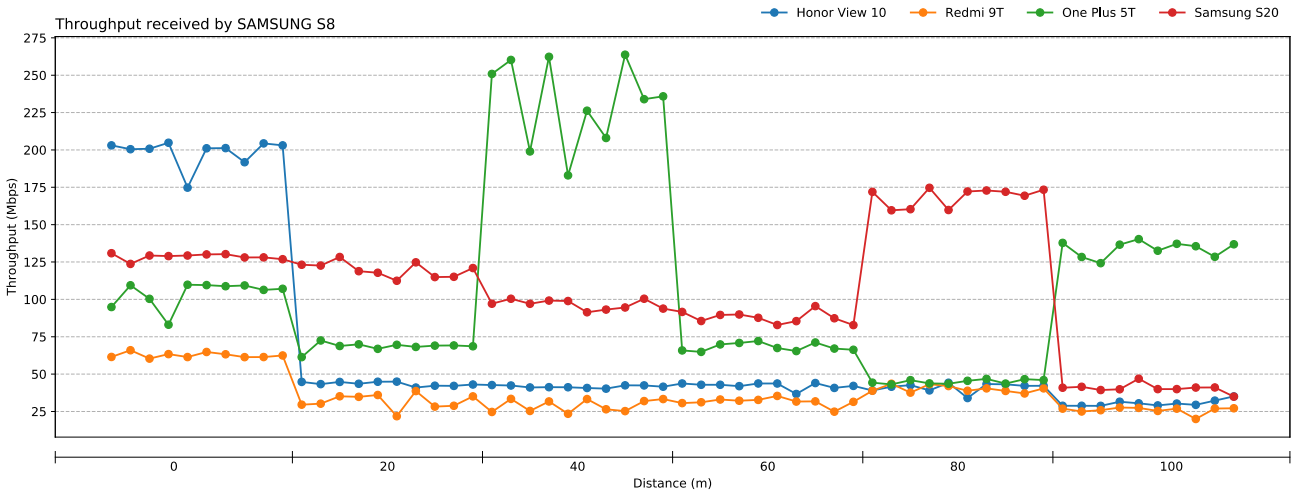

Figure 10: Throughput when the Samsung S8 is the receiver

Throughput received by SAMSUNG S20

Finally, we have the Samsung S20 (Figure 11), which, if we compare it with the throughput obtained using the One Plus 5T as a receiver (Figure 8), tends to behave similarly. Both have throughput regularly above 75 Mbps, as well as peaks above 700 Mbps at 0 meters. In addition, they also have a throughput drop at 60 meters. Finally, it can be observed that the throughput of all transmitters from 20 meters maintains a similar behaviour with the exception of the Samsung S8, which has a more consistent throughput regardless of the distance at which it is located.

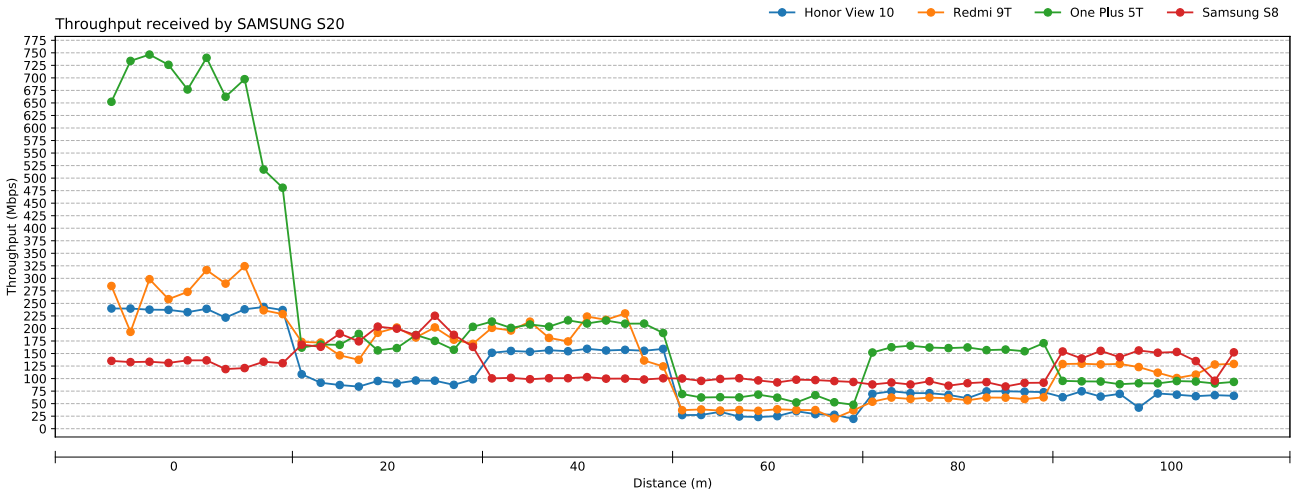

Figure 11: Throughput when the Samsung S20 is the receiver

3 IoT MCM Communications

3.1 Background

The radio connectivity has been the major issue when considering MCM communications and there are several options focused primarily on cellular connectivity, Wi-Fi connectivity or customized/proprietary private-network connectivity. The availability of spectrum is one of the main showstoppers and over time there have been solutions spanning different frequencies. Some technologies take advantage of licensed and license-exempt spectrums to include cellular IoT, private LTE, private 5G, Wi-Fi and Low Power Wide Area (LPWA) IoT networks such as LoRaWAN®.

The connectivity of MCM networks may vary depending on the implementation conditions and their usage scenario taking into account the following requirements:

1. Coverage – this is one of the limiting requirements depending the coverage should be only short-range communications of a few meters, but also a long-range coverage of a few kilometres in urban areas and over 10 km in rural settings;
2. Reliability – this depends on the use case and scenario of the MCM communications depending the objective is to transmit a certain amount of traffic within a predetermined time duration with high success probability;
3. Latency – the service or application where MCM is used for determines this requirement;
4. Throughput – there is huge difference whether the MCM are intended to communicate a few kbps or Mbps;
5. Battery life – this requirement is important considering whether the MCM are in areas with little or no energy supply, whether the devices are continuously or temporarily connected to energy supply.
6. Device cost/complexity – aimed at enabling cost-efficient MCM network with many devices or a few very complex devices for high precision operations.

Based on these requirements the connectivity technology has to be selected since there is no single solution that can fulfil all the requirements. The LPWA applications are used for long-range, low power and low-cost use cases with relaxed throughput requirements. On the other hand, the cellular-IoT (C-IoT) technologies that 3GPP has been developing in the past years are designed using licensed frequencies over different technologies such as EC-GSM-IoT, LTE-M and NB-IoT, targeting LWPA. With the next generation of mobile networks i.e., 5G a new set of NR-based feature sets like NR Industrial IoT (IIoT) has been proposed in 3GPP.

3.2 Common MCM Technologies

Other C-IoT technologies non-3GPP has been designed as proprietary alternative options such as LoRa, Sigfox for large range and Zigbee and Bluetooth for lower range.

3.2.1 Short Range

Bluetooth LE: Bluetooth LE is a Personal Area Network (PAN) and its purpose is to connect to devices near a user. Bluetooth LE has a relatively short range of several tenths of meters, but it also has a significantly high data rate. Traditional Bluetooth had data rates varying between 1 to 3 Mbps, whilst Bluetooth LE data rate is 1 Mbps for short bursts. It is capable of switching to sleep mode in between those bursts allowing decreased power consumption. Bluetooth LE is also supported by many operating systems, including Android, iOS, Windows 8/10, and OS X. If a user has a smartphone and wants to connect to a device, Bluetooth LE makes that possible. However, Bluetooth isn't a great choice for high-density nodes or long-range applications. Bluetooth LE is ideal for someone travelling through a group of connected things in a defined space [27].

Zigbee: ZigBee is a mesh network protocol designed to carry small amounts of data across distances of several hundreds of meters. It is built to run on a mesh topology, meaning information from a single sensor node can travel across a group of modes until the transmission reaches the gateway. ZigBee is a Local Area Network (LAN), so unlike Bluetooth, it connects to devices that need a wider range. Despite ZigBee's wider range, it is still fairly limited and isn't the best choice for highly instrumented installations, due to the relatively low throughput. ZigBee networks have higher latencies, causing bottlenecks when multiple nodes try to send information through the same node to get to the gateway. Hence, this technology is not favoured when there's a high density of nodes. Moreover, ZigBee faces a lot of challenges when the link budget is highly variable, like mobile nodes or parking sensors [28].

Table 3: Bluetooth LE vs Zigbee [29]

	Bluetooth (LE)	ZigBee
Network Type	Personal area network (PAN), which supports few nodes	Local area network (LAN), which supports many nodes
Range	77 meters	291 meters
OS	Android, iOS, Windows 8, OS X	Not currently compatible
Topology	Mesh and star	Mesh only
Throughput	270 kbps	250 kbps
Modulation	Frequency-hopping spread spectrum (FHSS)	Direct-sequence spread spectrum (DSSS)
Transmit Power	10 mW	100 mW

3.2.2 Long Range

LoRa: LoRa is a non-cellular modulation technology for LoRaWAN. LoRa and LoRaWAN are not interchangeable since LoRaWAN is the standard protocol for WAN communications and LoRa is used as a wide area network technology. LoRa is used primarily in two ways; One is LoRaWAN, which has been deployed mostly in Europe and has a very small message capacity, as low as 12 bytes. The second way is the Symphony Link, product of Link Labs. Symphony Link is a wireless system built on LoRa technology that is designed to overcome the limitations of a LoRaWAN system. It is often included as a component of more complex LoRa networking solutions, mostly in the U.S. and Canada, and is designed for industrial applications [30]. LoRa represents a good radio network for IoT solutions and has better link budgets than other comparable radio technologies. However, in most cases, if one wants to connect to LoRaWAN networks, one needs to deploy his own network gateway. Although this might seem like a disadvantage, it actually makes LoRa a good alternative to WiFi for low power devices that need to be connected throughout a building, like a factory or a hospital. This technology is one of the few that can be used as a "do-it yourself" technology; any company can build and use their own connected device wherever they can put up the gateway. LoRa has several advantages such as being able to set up and manage your own network, it is capable to achieve command-and-control functionality due to the symmetric link. Additionally, LoRa devices work well when they are in motion, which makes them useful for outdoor asset tracking, whilst LoRa devices have long battery life. On the downside, LoRa has relatively low data rates and a relatively long latency time [31].

Sigfox: Sigfox is the company that awoke the world to the potential for IOT devices to use very low bandwidth connections. Sigfox has probably the lowest cost radio modules available in the market at less than $5 compared to $10-12 for other technologies. Sigfox is designed for uplink only although limited downlink is possible. Sigfox is an end-to-end network and technology player however, it has not deployed significant networks and is struggling as a company. Sigfox consumes a low amount of power, works well for simple devices that transmit infrequently, because it sends very small amounts of data very slowly. Additionally, it supports a wide coverage area. Communication is better headed up from the endpoint to the base station. It has bidirectional functionality, but its capacity from the base station back to the endpoint is constrained, and you'll have less link budget going down than going up. Mobility is difficult with Sigfox devices [32].

3.2.3 Cellular IoT

The cellular IoT has been a work item in 3GPP standards for several releases of the specifications. The Extended Coverage Global System for Mobile communications in the context of IoT (EC-GSM-IoT) was firstly introduced by the 3GPP in Release 13 as an EGPRS-based LPWA technology. The EC-GSM-IoT was designed as long-range, long battery life and low complexity system able to coexist with the existing mobile networks. The EC-GSM uses four GSM frequency bands (850, 900, 1800 and 1900 MHz). EC-GSM is improving the coverage by 20 dB over EGPRS, LTE-grade security, and power-efficient operation. EC-GSM is the first cellular IoT technology that was designed to support a huge number of devices (i.e., over 50.000 per cell) with the device identification and security associated with a cellular technology.

MCM design worked continued with LTE/4G part of 3GPP standards Release 13 that designed two variants of MCM solutions to support large-scale IoT deployments and a reduction in the device complexity. The first one is LTE-M (also known as eMTC) and the second is NB-IoT. The first one operates in regular LTE deployments, using the smallest possible channel size (1.4 MHz). The NB-IOT was designed to operate in a very small 180 kHz channel size, which allowed it to be deployed in standalone mode (typically reusing GSM channels), in regular LTE bands, or within LTE guard bands. The LTE-M uses normal LTE deployment so has higher capacity and supports mobility, VoLTE, and a data rate up to 1 Mbps. On other hand, NB-IoT is limited to 30 kbps but achieves better coverage and lower power consumption.

NarrowBand IoT: NB-IoT is an initiative by the Third Generation Partnership Project (3GPP), the organization behind the standardization of cellular systems, to address the needs of very low data rate devices that need to connect to mobile networks, often powered by batteries. As a cellular standard, the goal of NB-IoT is to standardize IoT devices to be interoperable and more reliable. Because NB-IoT is a cellular-grade wireless technology that uses OFDM modulation, the chips are more complex, but the link budgets are better. That means users get the high-performance level associated with cellular connections, but at the cost of more complexity and greater power consumption. NB-IoT is meant to be used to send and receive small amounts of data—a few tens or hundreds of bytes per day generated by low data-producing IoT devices. It is message-based, similar to Sigfox and LoRa, but with a much faster modulation rate that can handle a lot more data than those technologies. But NB-IoT is not an IP-based communication protocol like LTE-M. You can't actually connect to an IP network and expect to use it as you would with a smartphone. It was made for simple IoT applications and is more power efficient than LTE-M but designed for more infrequent communication purposes. The coverage is predicted to be very good and since NB-IoT devices rely on 4G coverage, they would work well indoors and in dense urban areas. It has faster response times than LoRa and can guarantee a better quality of service. It is difficult to implement Firmware-Over-The-Air (FOTA) or file transfers. Some of the design specifications for NB-IoT make it such that sending larger amounts of data down to a device is hard. Network and tower handoffs will be a problem, so NB-IoT is best suited for primarily static assets, like meters and sensors in a fixed location, rather than roaming assets [33].

Table 4: LoRaWAN vs Sigfox vs NB-IoT [34]

Technical Data	LoRaWAN	Sigfox (EU)	NB-IoT
Technology	Proprietary	Proprietary	Open standard
Licensed Spectrum	No	No	Yes
Max data rate (gross)	5.47 kbit/s (SF7)	0.1 kbit/s	27 kbit/s
Worst case data rate (-144 dB link budget)	0.297 kbit/s (SF12)	0.1 kbit/s	5-6 kbit/s
Max. payload length (data per message)	51 B (EU)/ 11 B (US)	12 B	1.000 B
Downlink capacity	Very low	Very low	Unlimited
Link budget/ max. path loss (Uplink)	141-146 dB	163 dB	164 dB
Link budget/ max. path loss (downlink)	151-156 dB	158 dB	164 dB

With 5G the design was focused on three different usage scenarios named enhanced Mobile Broadband (eMBB) communications, Ultra-Reliable and Low Latency Communication (URLLC) and massive Machine-Type Communication (mMTC). The next figure shows the evolution of MCM over the different 3GPP releases, where Release 15 is considered the starting point of the new 5G NR air interface that separates the three communication types indicated previously.

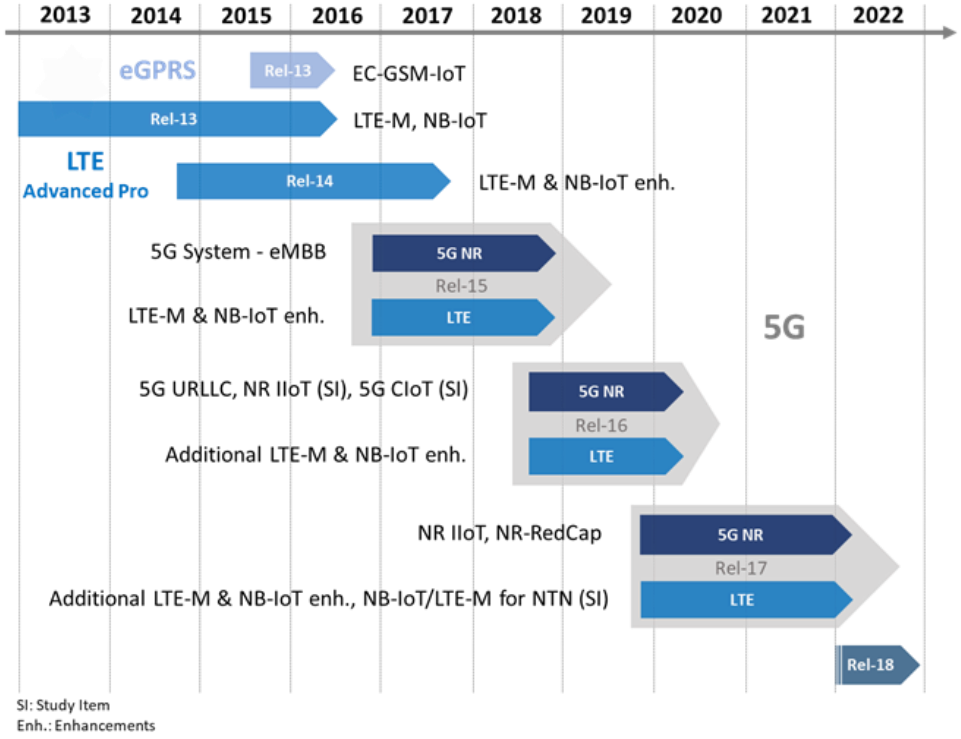

Figure 12: Evolution of C-IoT Technologies

The first type of communications supported by 5G NR is eMBB and it was defined in the first set of 5G specifications released by 3GPP as part of the 5G system (5GS). In the following Release 16, URLLC was defined that allows the NR radio interface to support both types of communications. The URLLC is aimed to support the new IoT applications part of machine-to-machine communication.

Moreover, 3GPP is continuing the specifications towards IoT and a new design work item related to NR Industrial Internet of Things (NR IIoT) is planned to be completed in the latest Release 17. The objective of this new type of communication is to support specifically industrial applications related to factory automation (i.e., logistics, sensor networks, robotics, augmented reality) with an integrated solution of NR eMBB and URLLC to support high transmission reliability and performance.

In addition, Time Sensitive Networking (TSN) is a key enabler for NR IIoT. It encompasses a set of standards identified by the IEEE 802 that enables Ethernet wired networks to ensure Quality of Service (QoS) features for time-sensitive traffic and critical-data applications, in order to provide deterministic transmissions by synchronizing various equipment components to a Grand Master (GM) clock. Moreover, with TSN the devices can utilize both internet protocol (IP) packets or Ethernet frames directly over the radio interface. The 5G system will be able to transport Ethernet frames by requesting PDU session type.

3GPP continues its work on MCM communications for the next releases. Thus, 3GPP has a new work item on NR Reduced Capability (NR-RedCap) devices and LTE-M/NB-IoT over Non-Terrestrial Networks (NTN). The NR IIoT benefits from the features of eMBB and URLLC but the objective is that NR-RedCap devices are ranging from LTE Cat-1 to Cat-4 in terms of power consumption and capabilities.

3.3 IoT-NGIN MCM Communication Technology Development

The usage of network slicing allows the allocation of network resources to selected devices. This allows to isolate MCM communications from other traffic and guarantee specific QoS to the devices associated with the slice. Network slicing is the concept of 5G that enables the creation of independent instances of 5G networks within a 5G system sharing the available resources in the Radio Access, Transport and core network functions. The industrial devices can be grouped and assigned to different network instances or slices and will be allocated different resources to isolate their traffic and might get different QoS levels compared to other slices.

The Network Slice Management System (NSMS) specified in 3GPP interacts with 5G RAN, Transport and Core network functions. The NSMS design follows the architecture defined in 3GPP TS 28.531 [35] that consists of an external Application Function (AF) that is provided as Graphical User Interface (GUI) to interact with 5G internal Network slice capability management server (Network Slice Management Service (NSMS)). Internally, the NSMS must deliver end to end resource allocation that would be managed through different vendor specific functions provided to manage the 5GS. 3GPP has defined in TS 28.531 the concept of slice subnet which is considered a different segment of the end-to-end system e.g., RAN subnet, Transport subnet, Core Network subnet. The NSMS will interact with the different slice subnets to allocate the required resources for creating the end-to-end network slice. The slices will optimize the communication of MCM with different QoS assigned to each slice.

The NSMS has been implemented on IoT-NGIN including the GUI shown in next Figure 13. The NSMS is implemented following the 3GPP specifications [TS 28.531] [35] to manage the RAN through the RAN-NSSMS (Network Subnet Slice Management System) to check and reserve radio resources for the network slice. The transport network resources will be managed through the TS-NSSMS (Network Subnet Slice Management System) and the NSMS will allocate core network functions for each slice with the CN-NSSMS. However, all these 5G complexities would be hidden through the OT application with the GUI for easy management of network slices.

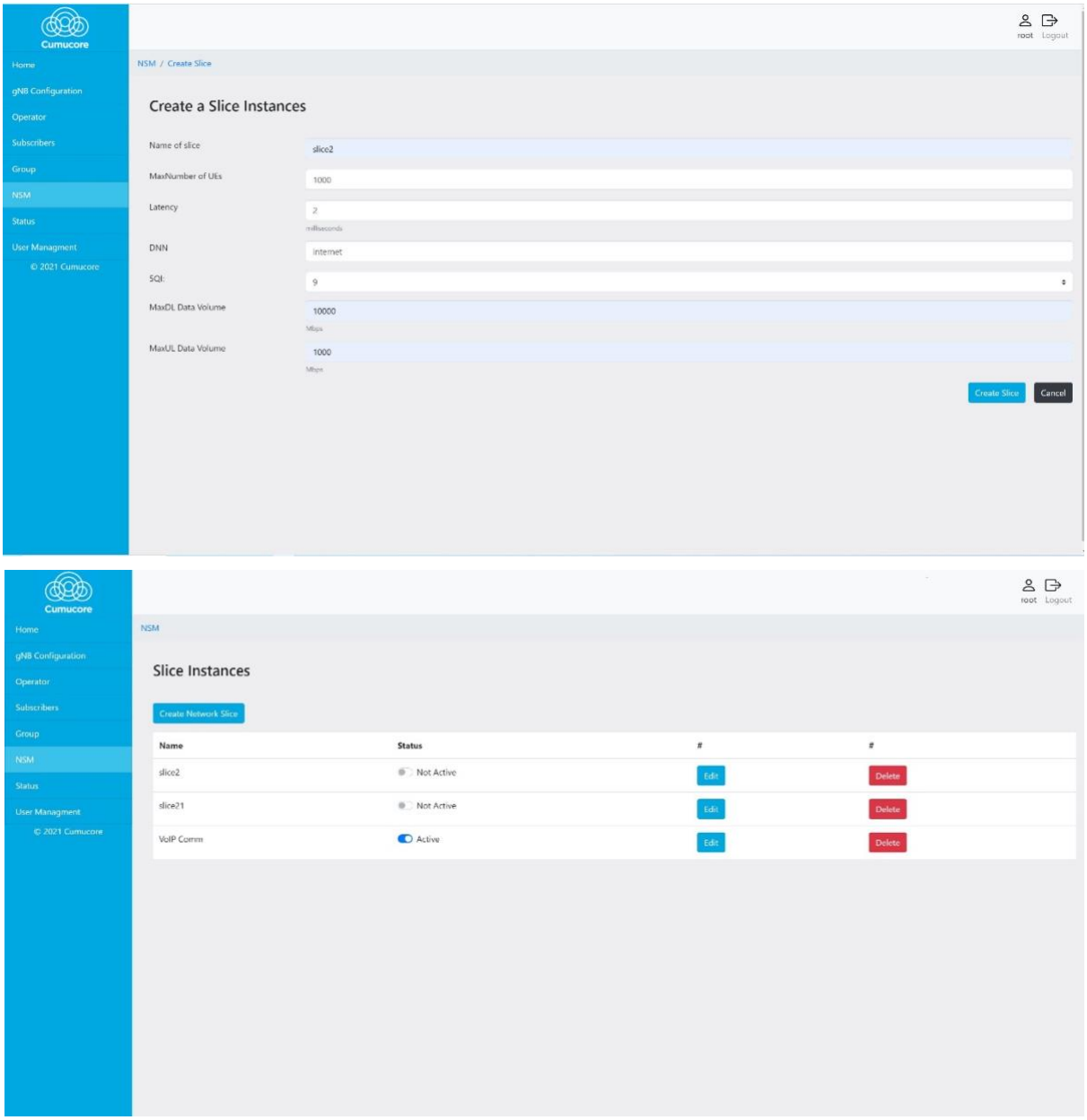

Figure 13: Create network slice

The implementation of the NSMS following the 3GPP architecture consists of the modules shown in Figure 14. The NSMS can be made accessible from external applications through Network Exposure function (NEF) as shown in the next figure 14. The slice metadata will be stored in dedicated Databases accessible through internal API (i.e., CNC) by all the network functions required for managing the slices.

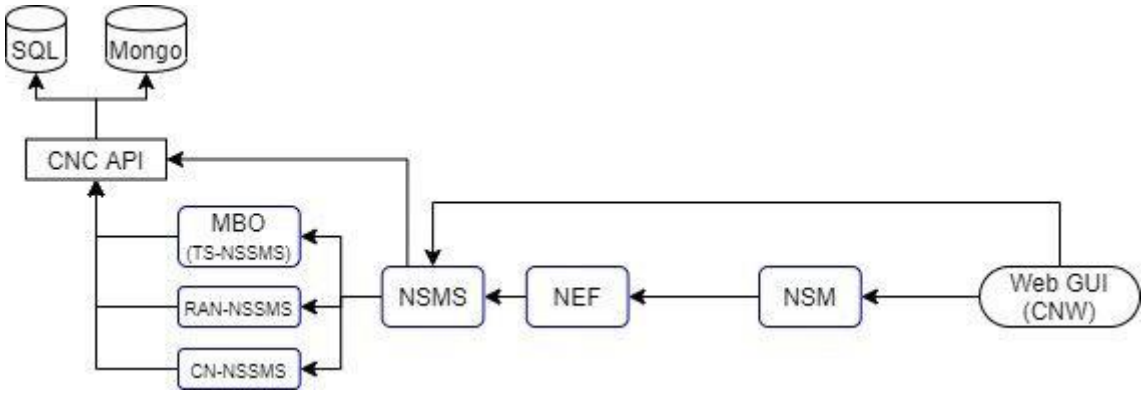

Figure 14: NSM architecture and functional modules

The NSMS provides an interface to create, activate, terminate slices in the network and perform feasibility checks before creating new slices. The NSMS will interact with different modules to create, delete, modify the network slices. Following are the different subcomponents required for managing the slices at different parts of the end-to-end system. A slice subnet considers different segment of the end-to-end system e.g., RAN subnet, Transport subnet, Core subnet. Thus, each subnet is managed by its own module that might be vendor specific. The Transport NSMS (TS-NSMS) will handle the interaction with the physical network to check available resources and set new policies or traffic priorities based on the slice requirements. The RAN NSMS (RAN-NSSMS) is the subnet module that will handle the network resources in the RAN and will check available configuration on the gNBs to confirm the allocation of end-to-end network slices including RAN, Transport and Core. The Core Network (CN) NSMS will take care of allocating the necessary processing resources to deploy the network functions allocated for each slice. Following are the specific interaction between NSMS and the different subnet components.

NSMS ->TS-NSMS

NSMS will interact with the so-called Transport Network (TN) manager TS-NSSMS that is checking the available transport capacity in the backhaul switches. The NSMS sends the transport network related requirements (e.g., external connection point, latency and bandwidth) to the TN Manager that reconfigures the TN accordingly (TS 28.531 section 5.1.1).

NSMS->RAN-NSMS

NSMS will interact with the other network or radio orchestrators RAN-NSSMS to collect or configure the available resources on RAN subnets for creating a new slice. NSMS will interact with Radio Resource Manager (RRM) to configure radio cells for the slice. The RAN-NSSMS will check the S-NSSAI (Single–Network Slice Selection Assistance Information) that has been configured in the gNB. The S-NSSAI is used to uniquely identify a Network Slice.

NSMS->CN-NSSMS

NSMS will interact with the Core Network Function Management Service CN-NSSMS or Network Function Virtualization Orchestrator (NFVO) to check available NF to be sued by the network slices or the CN-NSSMS can be used also to allocate additional network functions required for the slice. The CN-NSSMS can check the available NF registered in NRF based on supported TA, DNN and available capacity.

4 IoT-NGIN 5G Resource Management API

To make it easier for vertical sector customers to use 5G, we propose a 5G API to hide the complexity and make it easier for vertical sector customers to use the new functionalities, without the need to involve a mobile operator. Especially with regards to device management, there is no open API existing which provides these features.

This chapter will provide an overview of the work on the IoT-NGIN 5G Resource Management API. First, the requirements are grouped according to their exposure categories. Various standards were investigated to see which exposure categories and functions do already exist and which functions are not yet available. The mapping of the IoT-NGIN 5G Resource Management API requirements to the exposure categories and existing standards is provided in section 4.2.

The API architecture is shown and explained in section 4.3.

Also, 5G resource and capabilities that can be exposed through 5G API to IoT applications were chosen and grouped in three categories that cover the requirements of the IoT-NGIN 5G Resource Management API. For more details about general exposure categories, 5G resource and capabilities and standard specifications, please see section 4.4.

4.1 Background

An Application Programming Interface (API) in general is a kind of interface that software presents to other programs to provide functionality for that other software on the programming language abstraction level. In today's software landscape which is dominated by web- and microservices, these are often implemented by means of web technologies such as HTML and JSON. In that case, the software exposing the API includes a webserver, to which a client software can send the API calls as an HTML request and gets a corresponding answer also in the form of an HTML response. The software architecture style for such stateless web APIs is today known as Representational State Transfer (REST).

The document describing an API is known as an API specification. With the growing number of (publicly) available web APIs, the open OpenAPI Specification [36] for describing these APIs has emerged as the de-facto standard specification format. Originally developed in the Swagger project [37], it was formerly also known as swagger specification. Whilst the specification was transferred to the OpenAPI initiative, it should be noted that the Swagger project itself continues to provide numerous software projects to assist with the creation, editing and handling of such specifications in general.

4.2 Requirements

This chapter summarises the relationship between the requirements of the IoT-NGIN 5G Resource Management API and Open Standards. The following requirements (R1-R5) were identified:

[R1] 5G network connectivity including device management should be provided;

[R2] Initiation of microservices should be possible;

[R3] Decision on whether to have local or offloaded microservice execution should be possible;

[R4] Service migration execution should be possible;

[R5] Control of 5G resources (network slicing) should be possible.

Each of the requirements implies utilising 5G APIs enhancing the ease of use of 5G communication resources for IoT applications. This study recommends the usage of standardised 5G APIs. In this way, IoT applications will be able to interact with the underlying communications infrastructure utilising unified interfaces regardless of communications infrastructure vendor implementation.

The standard specifications that describe 5G resources and capabilities relevant to the requirements listed above were identified. They are grouped per requirements and listed in Table 5. Furthermore, the complete list of 5G resources and capabilities described in these standard specifications is listed in Annex 1and Annex 2.

Table 5: General mapping of the task requirements to the open standards

IoT-NGIN 5G Resource Management API Requirements	5G Exposure Categories	Open Standards
R1	Network connectivity and mobility capabilities exposure	3GPP: 23.682, 29.122, 23.502, 29.522, 23.434, 29.549
R2 – R4	Operational management capabilities and cloud infrastructure resources exposure	TMF: TMF641, TMF645, TMF638, TMF633, TMF623, TMF656 3GPP: 28.531, 32.615 ETSI: GS NFV-SOL 003, GS NFV-SOL 005 Kubernetes API
R5	Operational management capabilities	TMF: TMF641, TMF645, TMF638, TMF633, TMF623, TMF656 3GPP: 28.531, 32.615 ETSI: GS NFV-SOL 003, GS NFV-SOL 005

1) 3GPP – 3rd Generation Partnership Project

2) TMF – Tele Management Forum

3) ETSI – European Telecommunications Standards Institute

4.3 Architecture

The previous section has listed the requirements for the API. Changing from the requirements perspective to a functional view, we have identified the following four groups of capabilities for the IoT-NGIN 5G Resource Management API:

- 5G Connectivity and Device Management;
- Network Slice Management;
- Microservice Management;
- Service Migration.

The "*5G Connectivity and Device Management*" group encloses 5G capabilities such as device provisioning, device connectivity monitoring and management, device group management etc. This category is related to the first requirement "[R1] 5G network connectivity including device management".

The "*Network Slice Management*" features are mainly focused on the creation, configuration and deletion of network slices, where each network slice is formed of radio, cloud, and network parts. The requirement "[R5] Control of 5G resources" is handled in this part of the API.

"*Microservice Management*" provides functionalities for IoT services running in the edge-cloud. This includes management functionalities such as the creation or termination of new services, the management of existing services in terms of resource allocation (such as CPU or Memory) as well as the monitoring of the parameters of the services (e.g., status, resource consumption). These functionalities do not only fulfil requirement "[R2] Initiation of microservices" but do also provide the required information to support "[R3] Decision on whether to have local or offloaded microservice execution". As the two most relevant frameworks for microservices creation are Kubernetes and OpenStack [38], this part of the API will mainly interface with their corresponding APIs. Additionally, the network service instantiation and lifecycle management tasks are performed through OpenSource MANO (OSM) APIs.

The last group "*Service Migration*" enables a supervisory software or an IoT application directly to intelligently relocate an IoT service to either other nodes within the same edge-cloud or even to other nodes at different physical locations. Thus, it provides the necessary functionalities for [R4] Service migration execution". Use cases could be a device-service combination, where the device can change location, but the corresponding service should always be running in close physical proximity to minimize the communication latency. To migrate a running instance of a service to another compute resource inside the same installation Kubernetes and OpenStack expose service migration functionalities that can be used. For migration outside one cloud installation, higher level orchestration such as OpenSource MANO [39] are required.

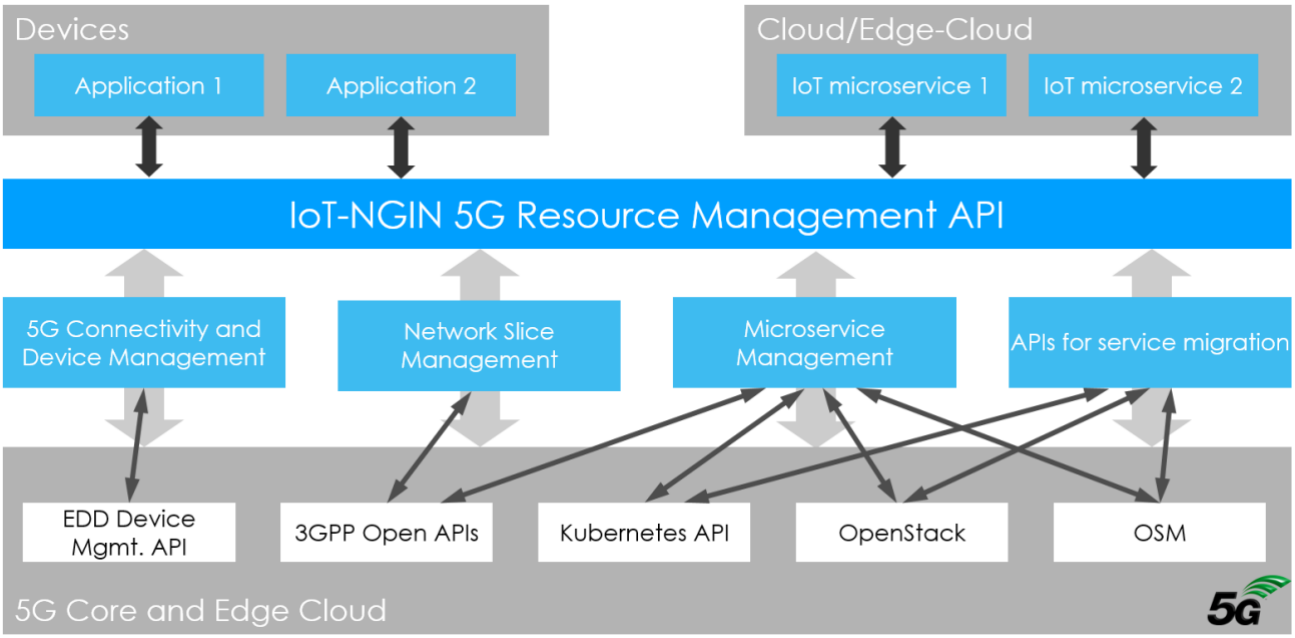

Figure 15: IoT-NGIN 5G Resource Management API

Figure 15 depicts the different layers of API communication that is necessary to enable interaction between an IoT application and the infrastructure. The envisioned API consumers are on the one hand the devices running software in the field and on the other hand the services running in a cloud/edge cloud environment. An API call that is utilizing a specific infrastructure function is handled by the "IoT-NGIN 5G Resources Management API". Depending on the functional category an API call is forwarded to the infrastructure specific API as a single or multiple calls to different API endpoints. The available infrastructure specific APIs depend on the specific implementation of the 5G network and can vary. The bottom layer of Figure 15 depicts infrastructure specific APIs. It should be noted that the selection is not exhaustive.

4.4 5G Exposure Categories

As described in the background chapter above, all standardised 5G resources and capabilities that can be exposed to the user were considered and grouped in general exposure categories. The categorisation was a complex task that was built on number of studies conducted in the project partner's organisations. The following three categories cover the requirements of the IoT-NGIN 5G Resource Management API, and they are the focus of this study:

- Network connectivity and mobility capabilities exposure;
- Operational management capabilities exposure;
- Cloud infrastructure resources exposure.

In each of the categories, the standards specifications that describe 5G resource and capabilities relevant to the requirements of the IoT-NGIN 5G Resource Management API were identified. Note that the 5G resource and capabilities exposure is a rapidly evolving area, and standard specifications are evolving continuously. Because of these reasons, the list of standard specifications specified in this document is not exhaustive.

4.4.1 Network connectivity and mobility capabilities exposure

The mobile network connectivity and mobility capabilities, traditionally available to the network operator solely, are exposed by a 5G network through the following functions that enable secure and developer-friendly access to network capabilities and services:

- Service Capability Exposure Function (SCEF);
- Network Exposure Function (NEF);
- Service Enabler Architecture Layer (SEAL).

The **Service Capability Exposure Function (SCEF)** provides means to securely expose the services and capabilities provided by 3GPP network interfaces through APIs to applications. It exposes network services and capabilities reducing the complexity for applications to access different 3GPP network services and capabilities. As of Release 16, 3GPP has specified the SCEF functions listed in Table 6.

The **Network Exposure Function (NEF)** is related to the 3GPP 5G architecture. This function provides means to securely expose the services and capabilities provided by 3GPP 5G network functions. NEF is the 5G core network equivalent of SCEF (LTE core network exposure function). The main difference is that 5G core network is based on Service Based

Architecture (SBA), which means that all communication between 5G core network functions and NEF is based on RESTful interfaces and no diameter interfaces like SCEF is using. Table 7 lists the NEF functions as defined in 3GPP standard specifications.

The **Service Enabler Architecture Layer (SEAL)** has been introduced in Release 16 of 3GPP to support applications from different industries (verticals). SEAL specifies a set of common services such as group management, configuration management, location management that can be used by different industry applications. SEAL is applicable to vertical applications using LTE or 5G radio access. Table 8 lists the SEAL services and their standard specifications.

Ericsson has developed a set of unique non-standard device management and device communication APIs on top of SCEF and NEF functions. They can support any type of device using the following protocols:

- Message Queue Telemetry Transport – MQTT;
- Constrained Application Protocol - CoAP (IP and Non-IP);
- Lightweight Machine-to-Machine – LwM2M (IP and Non-IP);
- Non-IP Data Delivery – NIDD [40]devices.

The private 5G technology provided by i2CAT also offers a set of non-standard network exposure functions, which are described here at a high level [41]:

- Public Land Mobile Network Identifier (PLMNID) management.
  - The proposed private 5G network technology allows to provision dedicated network slices. Each network slice is supported by a dedicate core network function that advertises a private network identifier, i.e., a private PLMNID. Private PLMNIDs must be unique within a given deployment but can be reused across deployments.
- SIM management.
  - Capability to add/remove users allowed to access a given network slice, where a user is identified by International Mobile Subscriber Identifier (IMSI). Only devices whose IMSI is provisioned in the core network function of a given network slice can connect to that slice.
- QoS management.
  - Capability to configure QoS settings on a per-slice basis. Configurable parameters include QoS Class Identifier (QCI) and Allowed Maximum Bit Rate (AMBR), on a per Access Point Name (APN) or per User Equipment (UE) basis.
- Cell management.
  - Capability to select the subset of 5G cells that need to be connected to a particular network slice. This capability can be used for example to restrict the availability of a given slice to a subset of cells, e.g., the indoor cells, whereas another slice may be available only on the outdoor cells.

4.4.2 Operational management capabilities exposure

Management functions handling the operational management aspects of exposed network capabilities can be exposed to the applications. This includes deployment, orchestration, monitoring, scaling, and the complete software development flow for the

exposed capabilities. Table 9 lists the operational management functions and the relevant standards. The private 5G solution provided by i2CAT offers network telemetry at the radio (per cell, per UE) and core network (e.g., per APN) level that can be used to feed monitoring dashboards or analytic engines that are key to simplify network management and operations.

4.4.3 Cloud infrastructure resources exposure

The Cloud Runtime Execution Environment (RTE) hosting telco and non-telco applications can be exposed. Apart from this, i2CAT's tool, Slicing & Orchestration Engine (SOE) offers a set of APIs, which are not only able to register and manage the Edge resources, but also performs the tasks related to service instantiation and management. More specifically, this tool provides:

a) The interaction with Virtual Infrastructure Manager (VIM) technologies such as OpenStack and Kubernetes through its main component, Slice Manager (SM), for the management of Edge infrastructures;
b) The interaction with Management and Orchestration (MANO) frameworks such as OSM through its other component, Multi-Tier Orchestrator (MTO), in order to deploy the network services and coordinate their Life Cycle Management (LCM) procedures.

Once a slice is requested SM breaks the request down and delegate it to other components of its framework by means of its REST-based APIs. The most relevant operational flows of SM to the Cloud infrastructure resources exposure category are as follows:

- Registration and management of Edge resources: it is regarded to the registration procedure of the main Edge resources performed through the interaction with VIMs such as OpenStack and Kubernetes. This operational flow also includes the following steps:
  - Store the registered compute resource objects in SM;
  - Edge resources related to APIs such as Kubernetes and OpenStack APIs to be reached from SM.
- Management of Edge partitions (chunks): this operational flow performs the registration and configuration procedures on the reserved Edge resources per slice named Edge compute chunk. In order to create a compute chunk, the following steps need to be completed:
  - Store a new compute chunk in SM;
  - Register different VIM accounts (OpenStack and Kubernetes) in OSM;
  - Create the related slice users and project for OpenStack and namespace for Kubernetes within OSM.
- Management of network slices: this task focuses on the commissioning procedures as well as on configuring the collection of chunks. Network slice instance management includes the following steps:
  - Create network slice instance in terms of reserved infrastructure resources including compute, access and network chunks;
  - Store network slice instance information in SM as a collection of chunks.

5 Conclusion and next steps

This document presented the activities carried out in WP2 in order to design the digital infrastructure and communication services necessary to support the IoT-NGIN solutions.

The D2D measurement tool, namely AtomD, will be released soon, and further experimentations will be done in order to characterize the D2D links and propose techniques for selecting the best relay strategy enabling coverage extension. At the time of writing this deliverable, we have run several experiments by varying the distance between the source and destination nodes, and by using devices of different brands. Our preliminary results, which focus on the communication throughput, reveal two major points. Firstly, the capacity of the link depends on the distance separating the source and the destination, but the decrease in throughput is not strictly decreasing. Secondly, different brands lead to different results.

IoT MCM communications will be further studied. The Network Slice Manager Service (NSMS) supporting Time-Sensitive Networking is under development and will be tested. The current design and implementation of the NSMS has been described. Next the performance of data communications using different devices will be measured to evaluate the impact in future MCM communications.

The concept and the rationale of an API in the environment of 5G and IoT was presented. During that work the different, already available, standards were evaluated, and the four key functional groups of the API were identified. Based on these four groups a general architecture was developed.

With this API architecture in place and the standards mapping presented in section 4.2, the next step is the creation of the specification document as an OpenAPI document. This specification then will undergo a feedback cycle with the living labs to ensure applicability in real-world scenarios. Once the specification is final, the implementation work will start.

The work on the secure edge cloud framework has so far focused on the development of the underlying technologies. As now working prototypes do exist and the technology continues to mature, the work on the framework will start at the beginning of next year. This implies the finalization of the architecture as well as the implementation and testing work.

6 References

[1] "Kubernetes Website," The Linux Foundation, [Online]. Available: https://kubernetes.io/. [Accessed 30 11 2021].

[2] The Linux Foundation, "KubeVirt," [Online]. Available: https://kubevirt.io/. [Accessed 11 30 2021].

[3] S. Lankes et al, "RustyHermit - A Rust-based, lightweight unikernel," [Online]. Available: https://github.com/hermitcore/rusty-hermit. [Accessed 30 11 2021].

[4] "Open Container Initiative Runtime Specification," [Online]. Available: https://github.com/opencontainers/runtime-spec. [Accessed 30 11 2021].

[5] S. R. Pokhrel, J. Ding, J. Park, O. Park and J. Choi, "Towards enabling critical mMTC: A review of URLLC within mMTC," *IEEE Access,* pp. Vol. 8, pp. 131796-131813, 2020.

[6] C. Bockelmann, N. Pratas, H. Nikopour, K. Au, T. Svensson, C. Stefanovic, P. Popovski, A. Dekorsy, "Massive Machine-type Communications in 5G: Physical and MAC-layer solutions," 2021.

[7] S. R. Pokhrel, J. Ding, J. Park, O. Park and J. Choi, "Towards enabling critical mMTC: A review of URLLC within mMTC," *IEEE Access,* pp. vol. 8, pp. 131796-131813, 2020.

[8] G. Fodor et al., "An overview of device-to-device communications technology components in METIS," *IEEE Access,* pp. vol. 4, pp. 3288-3299, 2016.

[9] S. R. Pokhrel, J. Ding, J. Park, O. Park and J. Choi, "Towards enabling critical mMTC: A review of URLLC within mMTC," *IEEE Access,* pp. vol. 8, pp. 131796-131813, 2020.

[10] Chakraborty, C., Rodrigues, J.J.C.P., "A Comprehensive Review on Device-to-Device Communication Paradigm: Trends, Challenges and Applications.," *Wireless Pers Commun 114,* p. 185–207, 2020.

[11] Malathy, S., Jayarajan, P., Hindia, M.H.D.N. et al., "Routing constraints in the device-to-device communication for beyond IoT 5G networks: a review," *Wireless Netw 27,* p. 3207–3231 , 2021.

[12] J. M. B. da Silva Jr, G. Fodor, T. F. Maciel, "Performance Analysis of Network-Assisted Two-Hop D2D Communications," *Globecom 2014 Workshop – Broadband Wireless Access,* 2014.

[13] Singh, D., & Ghosh, S. C., "Mobility-aware relay selection in 5G D2D communication using stochastic model," *IEEE Transactions on Vehicular Technology, 68(3),* p. 2837–2849, 2019.

[14] de Mello, M. O. M. C., Borges, V. C. M., Pinto, L. L., & Cardoso, K. V. , "Improving load

balancing, path length, and stability in low-cost wireless backhauls. Ad Hoc Networks," *Ad Hoc Networks, 48,* p. 16–28, 2016.

[15] Zhang, H., Song, L., & Zhang, Y. J., "Load balancing for 5G ultra-dense networks using device-to-device communications," *IEEE Transactions on Wireless Communications, 17(6),* p. 4039–4050, 2018.

[16] Mohamed, E. M., Elhalawany, B. M., Khallaf, H. S., Zareei, M., Zeb, A., & Abdelghany, M. A., "Relay probing for millimeter wave multi-hop D2D networks," *IEEE Access, 8,* p. 30560–30574, 2020.

[17] Al-Kharasani, N. M., Zukarnain, Z. A., Subramaniam, S. K., & Hanapi, Z. M., "An adaptive relay selection scheme for enhancing network stability in VANETs," *IEEE Access, 8,* p. 128757–128765, 2020.

[18] Basak, S., & Acharya, T., "On energy efficient secure routing in multi-hop underlay D2D communications for IoT applications," *Ad Hoc Networks, 108,* p. 102275, 2020.

[19] Malathy, S., Jayarajan, P., Hindia, M.H.D.N. et al. , "Routing constraints in the device-to-device communication for beyond IoT 5G networks: a review," *Wireless Netw 27,* p. 3207–3231 , 2021.

[20] Google, "Nearby, A platform for discovering and communicating with nearby devices," [Online]. Available: https://developers.google.com/nearby.

[21] G. Arena, "OnePlus 5t," [Online]. Available: https://www.gsmarena.com/oneplus_5t-8912.php.

[22] G. Arena, "Honor View 10," [Online]. Available: https://www.gsmarena.com/honor_view_10-8938.php.

[23] G. Arena, "Samsung S20," [Online]. Available: https://www.gsmarena.com/samsung_galaxy_s20_fe_5g-10377.php.

[24] G. Arena, "Samsung S8," [Online]. Available: https://www.gsmarena.com/samsung_galaxy_s8-8161.php.

[25] G. Arena, "Xiaomi Redmi 9T," [Online]. Available: https://www.gsmarena.com/xiaomi_redmi_9t-10670.php.

[26] I. O. f. Standardization, "IEC 80000-13:2008 Quantities and units — Part 13: Information science and technology," ISO, 2008.

[27] "Zigbee vs. Bluetooth: Choosing the Right Protocol for Your IoT Application," [Online]. Available: https://www.digi.com/blog/post/zigbee-vs-bluetooth-choosing-the-right-protocol. [Accessed 30 12 2021].

[28] "Difference between Bluetooth and Zigbee," [Online]. Available: https://www.geeksforgeeks.org/difference-between-bluetooth-and-zigbee/. [Accessed 30 11 2021].

[29] "A Bluetooth & ZigBee Comparison For IoT Applications," [Online]. Available: https://www.link-labs.com/blog/bluetooth-zigbee-comparison. [Accessed 30 11 2021].

[30] "LoRa Alliance," [Online]. Available: https://lora-alliance.org/ . [Accessed 30 11 2021].

[31] "NB-IoT vs. LoRa vs. Sigfox," [Online]. Available: https://www.link-labs.com/blog/nb-iot-vs-lora-vs-sigfox . [Accessed 30 11 2021].

[32] "SIGFOX," [Online]. Available: https://www.sigfox.com/en . [Accessed 30 11 2021].

[33] "Narrowband IoT," [Online]. Available: https://en.wikipedia.org/wiki/Narrowband_IoT . [Accessed 30 11 2021].

[34] "NB-IoT, LoRaWAN, Sigfox: An up-to-date comparison," [Online]. Available: https://iot.telekom.com/resource/blob/data/492968/e396f72b831b0602724ef71056af5045/mobile-iot-network-comparison-nb-iot-lorawan-sigfox.pdf. [Accessed 30 11 2021].

[35] "REF TS 28.531: 5G;Management and orchestration; Provisioning," [Online]. Available: https://www.etsi.org/deliver/etsi_ts/128500_128599/128531/16.06.00_60/ts_128531v160600p.pdf.

[36] "Open API Initiative," [Online]. Available: https://www.openapis.org/. [Accessed 30 11 2021].

[37] "Swagger Project," SMARTBEAR, [Online]. Available: https://swagger.io/. [Accessed 30 11 2021].

[38] OpenInfra Foundation, "OpenStack Website," [Online]. Available: https://www.openstack.org/. [Accessed 30 11 2021].

[39] "Open Source MANO Website," ETSI, [Online]. Available: https://osm.etsi.org/. [Accessed 30 11 2021].

[40] "Non-IP Data Delivery (NIDD) is non-IP based protocol used in LTE network for the communication with low-power IoT devices that allows small amounts of data to be transferred over the control plane".

[41] "The formal definition of the endpoints made available by the API will be included in subsequent deliverables.".

Annex 1 Network connectivity and mobility standard specifications

Table 6: GPP SCEF functions and the standard specifications

SCEF Functions Specified in 3GPP Standard Specifications 23.682 and 29.122
Device triggering procedures
Information storage
Security procedures
Group message delivery procedures
Monitoring procedures
High latency communications procedures
Procedure for Informing about Potential Network Issues
Procedure for resource management of background data transfer
Communication Pattern parameters provisioning procedure
Setting up an application server session with required QoS procedure
Change the chargeable party at session set-up or during the session procedure
Non-IP Data Delivery procedures
Packet flow descriptions management via SCEF
Procedure for MSISDN-less mobile originated SMS via T4
Procedure for enhanced coverage restriction control via SCEF
Procedures for accessing machine type communications-interworking function functionality via SCEF
Procedure for network parameter configuration via SCEF
Optimisation of radio access capability signalling information provisioning procedures

Table 7: 3GPP NEF functions and the standard specifications

NEF Functions Specified in 3GPP Standard Specifications 23.502 and 29.522
Procedures for monitoring
Procedures for device triggering
Procedures for resource management of background data transfer
Procedures for communication pattern parameters provisioning
Procedures for packet flow description management
Procedures for traffic Influence
Procedures for changing the chargeable party at session set up or during the session
Procedures for setting up an application function session with required QoS
Procedures for MSISDN-less mobile originated SMS
Procedures for network configuration parameters provisioning
Procedures for non-IP data delivery
Procedures for radio capabilities signalling optimisation parameter provisioning
Procedures for analytics information exposure
Procedures for 5G LAN parameter provisioning
Procedure for applying background data transfer policy
Procedures for enhanced coverage restriction control
Procedures for IPTV configuration
Procedures for location privacy indication parameters provisioning
Procedures for service specific parameter provisioning
Procedures for auto-configuration server configuration parameter provisioning
Procedures for mobile originated location request
Procedures for authentication and key management for applications
Procedures for time synchronization exposure
Time synchronization parameters provisioning
Procedures for edge configuration server address provisioning
Procedures for access and mobility management policy authorization

Table 8: 3GPP SEAL services and the standard specifications

SEAL Services Specified in 3GPP Standard Specifications 23.434 and 29.549
Location management
Group management
Configuration management
Identity management
Key management
Network resource management

Annex 2 Operational management capabilities standard specifications

Table 9: Operational management functions and the standard specifications

Operational Management Functions	Standard Specifications
Order Management	TMF641
Service Qualification	TMF645
Inventory Management	TMF638
Service Catalog	TMF633
SLA Management	TMF623
Service Problem	TMF656
Network Slice Orchestration	28.531
Application Configuration	32.615
CNF[1)]/CNA[2)] Lifecycle Management	ETSI GS NFV-SOL 003
CNF/CNA SW Package Management	ETSI GS NFV-SOL 005

1) *CNF - Cloud Native Function*
2) *CNA - Cloud Native Application*