ETSI TR 103 842 V1.1.1 (2025-02)

SmartM2M;
Demonstration of Performance Evaluation and Analysis for oneM2M Planning and Deployment

Reference
DTR/SmartM2M-103842

Keywords
evaluation, KPI, oneM2M, open source,
performance, scenarios, simulation

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI IPR online database.

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™**, **LTE™** and **5G™** logo are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

Introduction

ETSI TTF 019 aims are studying, analysing, evaluating, and simulating IoT application deployments on some oneM2M open-source implementations. The overall evaluation is conducted based on case studies, deployment model, and performance KPIs (Key Performance Index), all described in ETSI TR 103 839 [i.1] and ETSI TR 103 840 [i.2].

The object of the present document [i.4] is to describe and present different demonstrators build on top of the simulation tool and the profiler tool developed in ETSI TR 103 841 [i.3]. More precisely, the simulation tool is a OMNeT++ library implementing both the deployment model and the case studies as described in [i.1] and [i.2]. The profiler is a standalone software to be run together with a real open source oneM2M implementation to measure and record the impact on computer resources like memory or processor. Those real measures are used as input for the simulator.

The present document provides the configuration elements of three demonstrators (described in [i.1]), highlights classical behaviours in IoT system deployment and illustrates features of the simulation tools in terms of KPI analysis.

1 Scope

1.1 Context for the present document

The oneM2M standard is now mature: multiple deployments exist all over the world at both experimental and operational levels. The experimental deployments are conducted for multiple reasons:

- to evaluate the capabilities of the standard in terms of expressiveness, usability on specific equipment, connection with specific existing systems or performance evaluation;
- to provide a methodological study on a given set of "paradigmatic use cases";
- to measure KPIs, defined later in the present document, of implementations that are compliant with the oneM2M standard, available either freely or commercially.

Use cases are characterized in terms of chosen KPI: e.g. running time, memory usage, numerosity of oneM2M entities (e.g. AE, MN-CSE, CSE), data transfer volume and message latency and throughout needs. Using a selected set of available oneM2M CSE implementations [i.6], a simulation library or an ad hoc simulator is to be provided, offering the ability to evaluate and simulate the performance of the use cases and give crucial information/feedback to the general user of oneM2M to choose and tune their IoT applications based on oneM2M framework [i.5]. The results of this tool development and evaluations of the use cases will be the basis to generate other deliverables. The present document was developed in the context of ETSI TTF T019, set up to perform work on "Performance Evaluation and Analysis for oneM2M Planning and Deployment". Five elements were addressed sequentially:

- A collection of **use cases and derived requirements** were formally identified and defined. This work includes identification of relevant deployment scenarios. This phase of the work resulted in deliverable ETSI TR 103 839 [i.1]. The present document adopted the use case style and template from oneM2M with a minor modification to address some performances issues.
- The definition of **performance evaluation model**, with specification of procedures to assess the performance of oneM2M-based IoT platforms. This includes the identification and definition of KPIs necessary to assess the deployment. For those KPIs, provision of a formal description of the test campaign and the test results to be obtained. This phase of the work resulted in deliverable ETSI TS 103 840 [i.2].
- The creation of a **proof of concept** of a performance evaluation tool. This work also relies on a formal description of the identified deployment scenarios (single vertical domain & multiple vertical domains). This phase of the work resulted in deliverable ETSI TS 103 841 [i.3]
- A practical **demonstration and analysis** exercise putting the proposed tool to use, with a specific oneM2M implementation but aimed at being a blueprint for the adoption and re-use of the results of ETSI TR 103 839 [i.1], ETSI TS 103 840 [i.2]. This phase of the work resulted in the present document.
- The development of a set of **guidelines and best practices** documenting best practices and lessons learned as well as providing instructions for IoT solution topology, capacity provisioning, and expected performances that will gives crucial directives and information to designer and implementors. This phase of the work will result in the deliverable ETSI TR 103 843 [i.4].

1.2 Scope of the present document.

The present document shows example of use of the simulation tools developed in [i.3]. These three examples called *Smart Campus*, *eHealth monitoring* and *Traffic light* were chosen to illustrate behaviours specific to IoT systems, such as the use of large-scale multi-domain and heterogeneous environment, response time in asynchronous communications, but also how exploring the underlying mechanisms of oneM2M can help in solving temporal problems in certain cases.

The present document is structured as follows:

- Clauses 1 to 3 set the scene and provide references as well as definition of terms, symbols and abbreviations, which are used in the present document.
- Clause 4 describes the process to create a simulation.

- Clause 5 describes a first smart campus demonstrator as a multi-domain heterogeneous system.
- Clause 6 describes the eHealth monitoring demonstrator that mix event-triggered/Periodic event IoT system.
- Clause 7 describes the traffic light control and monitoring demonstrating the scalability of the architecture.
- Clause 8 provides the conclusions of this work.

Annex A provides a link to the code sources with the aim of result reproducibility.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TR 103 839: "SmartM2M; Scenarios for evaluation of oneM2M deployments".

[i.2] ETSI TS 103 840: "SmartM2M; Model for oneM2M Performances Evaluation".

[i.3] ETSI TR 103 841: "SmartM2M; oneM2M Performance Evaluation Tool (Proof of Concept)".

[i.4] ETSI TR 103 843: "SmartM2M; oneM2M deployment guidelines and best practices".

[i.5] oneM2M TS-0001 (V3.34.0): "Functional Architecture".

[i.6] oneM2M: - "List of oneM2M deployments".

[i.7] OMNeT++: "Discrete Event Simulator".

[i.8] Tools Deliverable ETSI repository .

NOTE: See Annex A.

[i.9] OMNeT++: "Installation Guide".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the following terms apply:

guidelines and good practices: methodological document that gives hints to deploy a oneM2M infrastructure

Key Performance Index (KPI): list of criteria to be measured on a system

oneM2M deployment: mapping of a IoT applications on a oneM2M infrastructure

performance evaluation: evaluation of temporal, data transfer volumetry, and scalability aspects of a system

platform evaluation tool: simulation environment that is used to calculate/demonstrate the performance of a system

profiler: monitoring tool measuring KPIs

real time constraints: dynamic constraints to be fulfilled related to time

single/multiple horizontal/vertical domains: interaction capability of many oneM2M infrastructures from different domains

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

ACP	Access Control Policy
ADN	Application Dedicated Node
AE	Application Entity
ARM	Advanced RISC Machine
ASN	Application Service Node
BACNET	Building Automation and Control Networks
BLE	Bluetooth Low Energy
CIN	Content Instance
CNT	Container
CoAP	Constrained Application Protocol
CPU	Central Process Unit
CRUD	Create, Read, Update, Delete
CSE	Common Service Entity
CSV	Comma Separated Values
ESA	Emergency Server Application
ETSI	European Telecommunications Standards Institute
GPRS	General Packet Radio Service
HTTP	Hyper Text Transfer Protocol
IDE	Integrated Development Environment
IN-CSE	Infrastructure Node - Common Services Entity
IoT	Internet of Things
JSON	JavaScript Object Notation
KPI	Key Performance Index
LoRa	Long Range
M2M	Machine-to-Machine
Mca	Reference Point for M2M Communication with AE
Mcc	Reference Point for M2M Communication with CSE
MIPS	Million Instructions Per Second
MN	Middle Node
MN-CSE	Middle Node - Common Services Entity
MQTT	Message Queue Telemetry Transport
NED	NEtwork Descriptor
OASd	oneM2M Application Scenario descriptor
OASD	oneM2M Application Scenario Descriptor
OCPd	oneM2M CSE Performance descriptor
OCPD	oneM2M CSE Performance Descriptor
OM2M	Eclipse OM2M - Open-Source platform for M2M communication
OSDd	oneM2M Solution Deployment descriptor
OSDD	oneM2M Solution Deployment Descriptor
PER	Packet Error Rate
PLA	Patient Local Application
PSA	Physician Server Application

RAM	Random Access Memory
RTT	Round Trip Time
SDK	Software Development Kit
SUB	Subscription
TCP	Transport Control Protocol
TR	Technical Report
TS	Technical Specification
Wi-SUN	Wireless-Smart Ubiquitous Network
YAML	Yet Another Markup Language

4 Process for simulation creation

The study of a specific use case based on a simulation approach follows a specific process:

- Hardware inputs: select a specific oneM2M stack and specific hardware to run this stack with the profiler to extract hardware usage for oneM2M resources or use predefined values and scale factor to approach hardware behaviour.
- Define the oneM2M architecture (IN-CSE, MN-CSE) and the performance of each CSE using the oneM2M CSE Performance descriptor (OCPd).
- Define the oneM2M applications with their sensors and actuators behaviour in terms of oneM2M resources using oneM2M Application Scenario descriptor (OASd).
- Define network and computer architecture using the oneM2M2 Solution Deployment descriptor (OSDd).
- Insert those descriptors into as input files for the OMNeT++ based simulator.
- Define the KPI and simulation metrics that should be extracted from simulation.
- Run the simulator in interactive or background mode.
- Analyse the results of the simulations using the CSV files produced by the simulator.

The following clauses walk through examples of this process using the sample scenarios defined in [i.1].

5 Use Case - Smart Campus

5.0 Foreword

Based on the use case described in clause 5 of [i.9], the IoT architecture is deployed inside the university campus with sensors, networks, servers. Several domains are covered: Smart spaces, Environment monitoring, Weather monitoring, Water management, Energy management (Figure 5.1). This scenario illustrates the capability of the simulator to mix several domains and technologies and its ability to get relevant results that could be compared to execution on real hosts and oneM2M stacks.

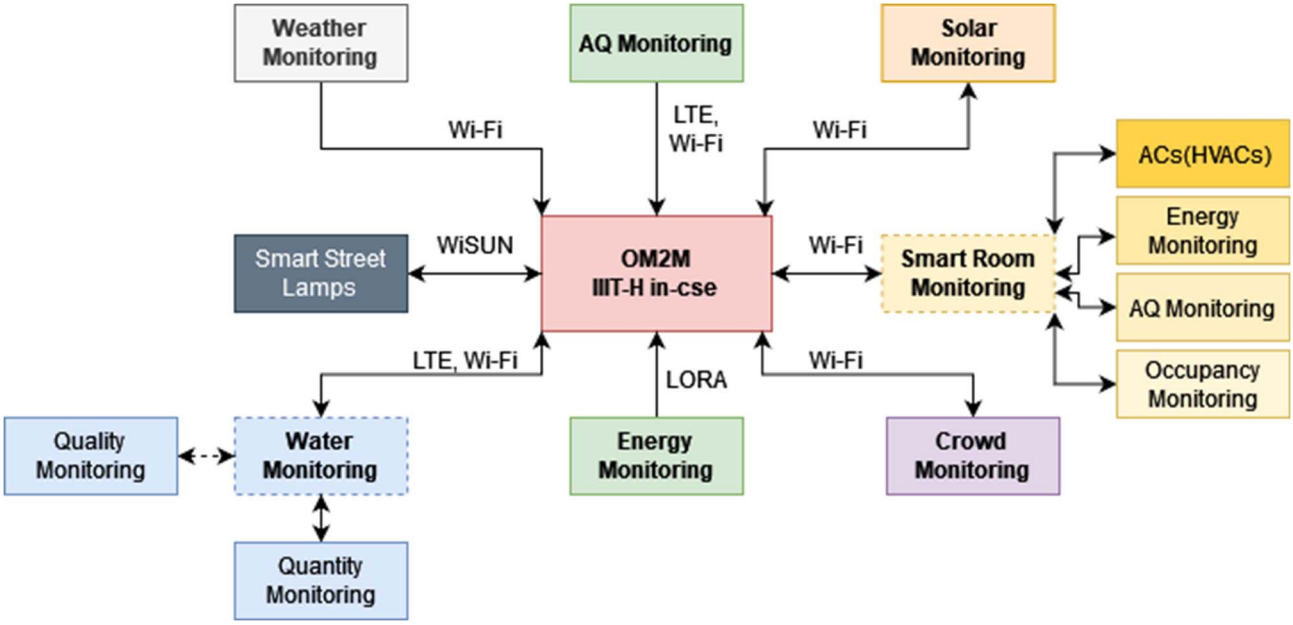

Figure 5.1: Architecture of a smart campus

5.1 Representation

5.1.1 High level view of the oneM2M Application Scenario descriptor (OASd)

OASd meta-model has been described in [i.2]. To model this scenario and map to the capabilities of the simulator, several CSEs have been created. Each category of sensors has been described in terms of number of sensors, probabilistic or deterministic time between two consecutive CIN generation and the data size of CIN creation messages. The links between sensors and the different CSEs have been expressed (Figure 5.2).

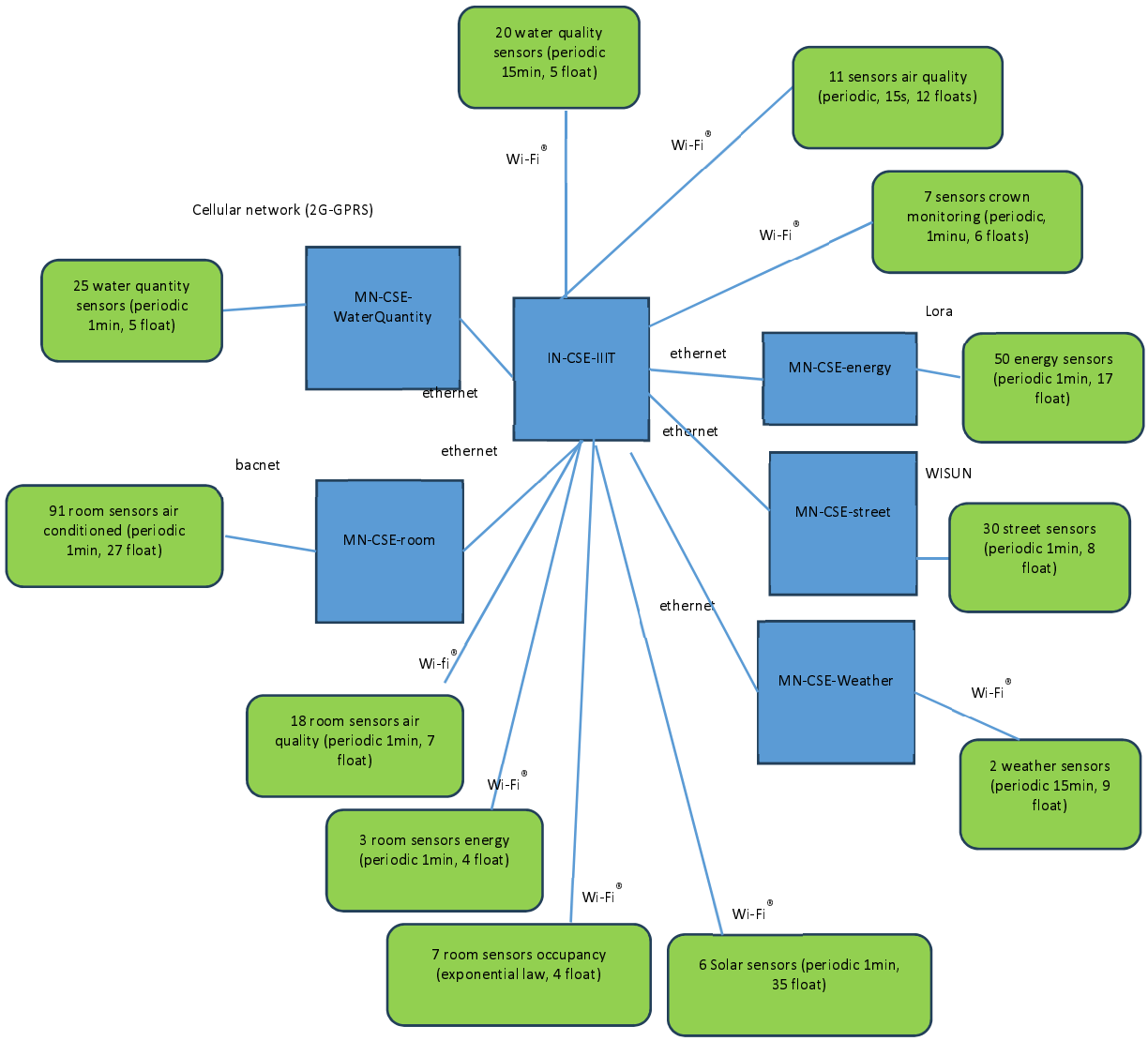

Figure 5.2: Smart campus application scenario description

The first step is to describe the characteristics of the sensor data. These parameters are settled in the OMNeT++ simulator in the *omnetpp.ini* file that collects the different parameters of the IoT system. The OASd part corresponding to the use case is the following. For example, the water quantity sensor send data every 60 s and the size of the message is 20 bytes. The schema for the OASd is provided in Annex B.

```
{
   "type":"SimulatedEvent",
   "eventDistribution":{
      "type":"Constant",
      "constant":60
   },
   "dataSizeDistribution":{
      "type":"Constant",
      "constant":20
   }
}
…
```

This JSON object is then provided as a value for the simulation parameter *cinGenerator* of the corresponding IoT Node:

```
Network.waterQuantitySensor[*].application[0].sensor[0].cinGenerator
```

The second step is to describe the connection between each node in the IoT system. The network topology is defined in *Network.ned* file. This file also includes some information related to the different nodes such IoT Nodes, CSE Nodes, and Server Nodes if any.

For example, in the Campus UseCase topology, the IoT solution contains a middle node CSE : mnWater. Its name is "mnWater-cse". Its parent CSE is the Infrastructure Node CSE (in-cse). The mnWater node has one network interface that is connected to an ethernet network. The IP address of the mnWater node is "10.0.1.1".

The IoT solution also contains 25 IoT nodes that host water quantity sensors. All these IoT nodes have one application that manages one sensor and no actuators. The name of the application is "appWaterQiuantity1" for the first IoT nodes, "appWaterQuantity2" for the second IoT node, and so on. The data generated by each sensor is stored within the corresponding AE : "AE_WaterQuantity_1", "AE_WaterQuantity_2", etc. Finally, each IoT node has one network interface with an IP address in the form "10.0.1.X" (with X starting at 2).

```
mnWater: CSENode {
  name = "mnWater-cse";
  cse.remoteCSE = "in-cse";
  cse.remoteCSEAddress = "10.0.0.1";
  nic[0].name = "eth0";
  nic[0].networkAddress = "10.0.1.1";
}
…
waterQuantitySensor[25]: IoTNode {
  appCount = 1;
  application[0].sensorCount = 1;
  application[0].actuatorCount = 0;
  application[0].name = "appWaterQuantity_" + string(index);
  application[0].sensor[0].name = "AE_WaterQuantity_" + string(index);

  application[0].remoteCSE = "mnWater-cse";
  application[0].remoteCSEAddress = "10.0.1.1";

  nic[0].name = "wan0";
  nic[0].networkAddress = "10.0.1." + string(index + 2);
}

roomACSensor[91]: IoTNode {
         appCount = 1;
         application[0].sensorCount = 1;
         application[0].actuatorCount = 0;
         application[0].name = "appRoomAC_" + string(index);
         application[0].sensor[0].name = "AE_RoomAC_" + string(index);

         application[0].remoteCSE= "mnRoom-cse";
         application[0].remoteCSEAddress = "10.0.2.1";

         nic[0].name = "eth0";
         nic[0].networkAddress = "10.0.2." + string(index + 2);
}
…
```

The overall file (*Network.ned*) can be found in the source code referenced in Annex A.

5.1.2 High-level view of the oneM2M2 CSE Performance descriptor (OCPd)

OCPd meta-model has been described in [i.2]. The stack used in the demonstrator is OM2M with these features for CSE (described in the *omnetpp.ini* file). The values given by the profiler are used as inputs. For example, for an AE resource, the creation operation takes an estimation of 509 instructions and uses 288 358 bytes in memory.

```
   "PerformanceDescriptor": {
      "name": "om2m",
      "product": "Eclipse OM2M",
      "version": "v1.1",
```

```
    "retargetingPerformances": {
       "processorUsage": 100.0,
       "memoryUsage": 10000.0
    },
    "resourcePerformances": {
       "AE": {
          "processorUsage": {
             "createOperation": 509,
             "deleteOperation": 720
          },
          "memoryUsage": {
             "createOperation": 288358,
             "deleteOperation": 419430
          },
```

This JSON object is then provided as a value for the simulation parameter *performanceDescriptor* of the corresponding CSE Node:

```
Network.mnWater.cse.performanceDescriptor
```

The overall file (*omnetpp.ini*) can be found in the source code referenced in Annex A.

5.1.3 High-level view of the oneM2M2 Solution Deployment descriptor (OSDd)

OSDd meta-model has been described in [i.2]. The IN-CSE-IIT, an infrastructure CSE is running on Server. This computer is based on an Intel® Core-i5-8400 2,80 GHz processor, with 8 GB of RAM. Several gateways have been created to host all middle node CSE: MN-CSE-WaterQuantity, MN-CSE-room, MN-CSE-energy, MN-CSE-street and MN-CSE-Weather. The gateways are based on Raspberry Pi 4 with an ARM-Cortex-A72-4, 1,5 GHz with 2 GB of RAM. All this equipment is connected through ethernet network at 1 GB/s (Figure 5.3).

The characteristics of specific access networks used in the IoT solution are :

- 2G network: latency between 300 to 1 000 ms, bandwidth of GPRS: 40 kbps, Bytes error between 10-3 to 10-4, the retransmission rate is between 1 to 5 %.
- Wi-SUN network: latency between 20 to 100 ms, bandwidth between 50 to 300 kbps, Bytes error between 10-5 to 10-6, the retransmission rate is between 1 to 3 %.
- LoRa network: latency between 200 to 2000 ms, bandwidth between 0,3 to 50 kbps, Bytes error between 10-4 to 10-6, the retransmission rate is between 1 to 10 %.
- BACNET network: latency between 10 to 100 ms, bandwidth 100 Mbps, Bytes error between 10-9 to 10-12, the retransmission rate is less than 1 %.

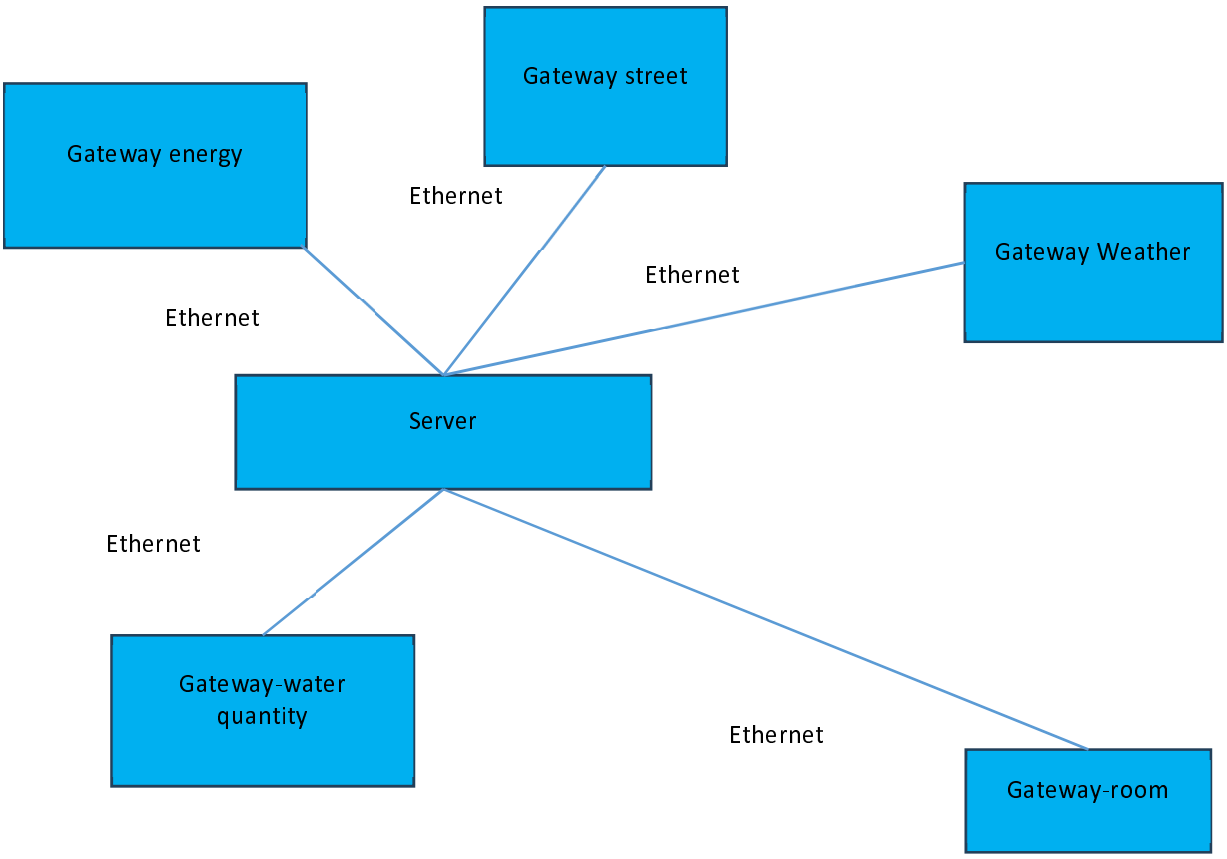

Figure 5.3: Smart campus computer and network description

The OSDD is expressed in *Network.ned* file. For example, the water Quantity Sensors are connected through a cellular network to the networkGateway. The networkGateway is connected to INTERNET using a fiber link.

```
connections:
  campusGateway.internetLink <--> FiberLink <--> INTERNET.link++;
  networkGateway1.internetLink <--> FiberLink <--> INTERNET.link++;
  …

  in.toNetwork[0] <--> EthernetLink <--> campusGateway.link++;
  mnWater.toNetwork[0] <--> EthernetLink <--> networkGateway1.link++;
  mnRoom.toNetwork[0] <--> EthernetLink <--> networkGateway2.link++;
  …

  for i=0..25-1 {
    waterQuantitySensor[i].toNetwork[0] <--> CellularLink <--> networkGateway1.link++;
  }
  for i=0..91-1 {
    roomACSensor[i].toNetwork[0] <--> EthernetLink <--> networkGateway2.link++;
  }
 …
```

The simulator builds the global representation of the system (Figure 5.4) with the equipment and their characteristics.

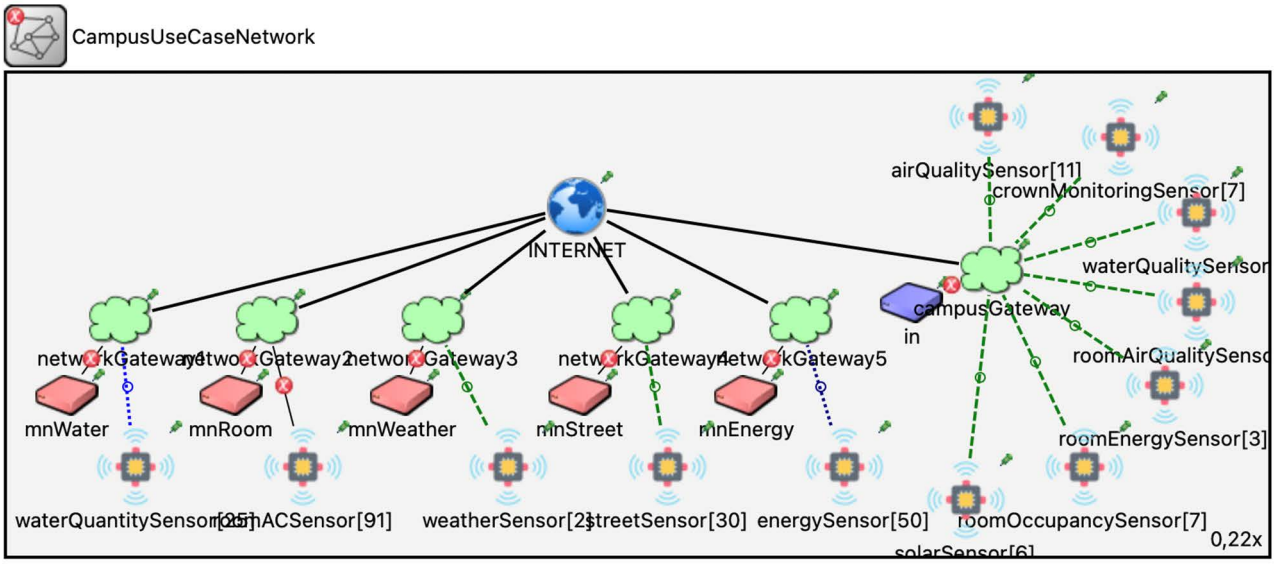

Figure 5.4: Smart Campus topology view in OMNeT++ simulator

5.2 Results

Based on the CSV files produced by the simulator, the KPI studied in the simulation is the Round-Trip Time (RTT) for the CIN messages generated by the IoT nodes sent to their remote CSE. The RTT is the time difference between the moment the IoT sends the oneM2M request and the moment it receives the corresponding response from the remote CSE. The remote CSE is either the IN-CSE or the MN-CSE depending on the adopted strategy for oneM2M resource storage location. Two simulation configurations have been conducted. In the first simulation, all content instances are stored on the IN-CSE using the retargeting mechanism of oneM2M. IoT nodes send their content instances creation messages to their domain MN-CSE and the MN-CSE retargets these oneM2M messages to the IN-CSE.

Figure 5.4 depicts the average, the minimum and the maximum RTT within each group of IoT sensors. For example, the average RTT for energy sensors is around 6 s. This can be explained by the fact that these sensors were connected through a LoRa network. However, the average RTT for air quality sensors is around 0,5 s since these sensors are connected through a Wi-Fi® network but also directly to the IN-CSE without an intermediate CSE. It worth noticing that the maximum RTT can go up to 2 s. This can be explained by the fact that the IN-CSE handles a lot of requests and thus some messages from these sensors can be delayed within the IN-CSE processing queues.

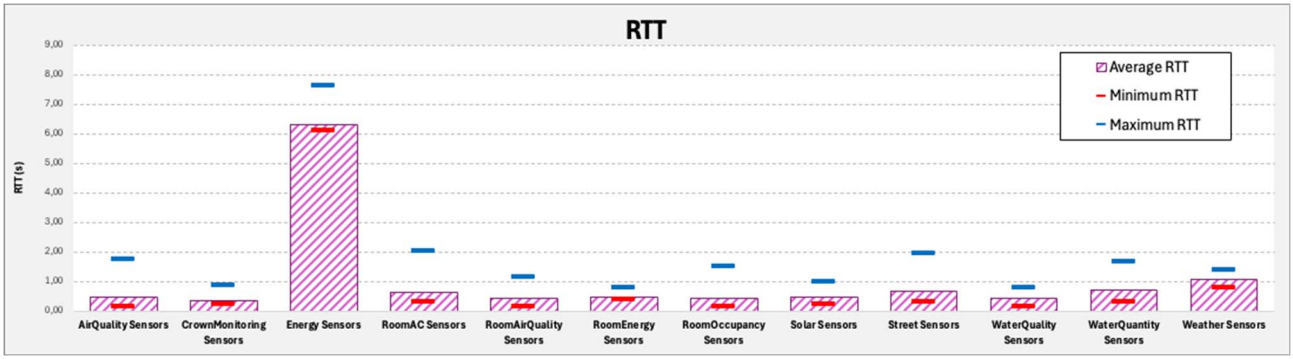

Figure 5.4: Simulated response time for the different families of sensors with centralization on IN-CSE

The simulator allows to modify easily the oneM2M strategy for resource location storage across the different CSE nodes. For example, one can simulate a distributed strategy where the sensors' data are stored on each MN of each specific domain of the smart campus rather than storing all these data on the IN-CSE. In this case, and as an example, the average RTT (Figure 5.5) of air quality sensors is 0,15 s (a 31 % decrease compared to the centralized strategy). It is worth noticing that either centralized or distributed strategy has no significant impact on the RTT for energy sensors since they are mostly impacted by the delay induced from the underlying network technology (LoRa network).

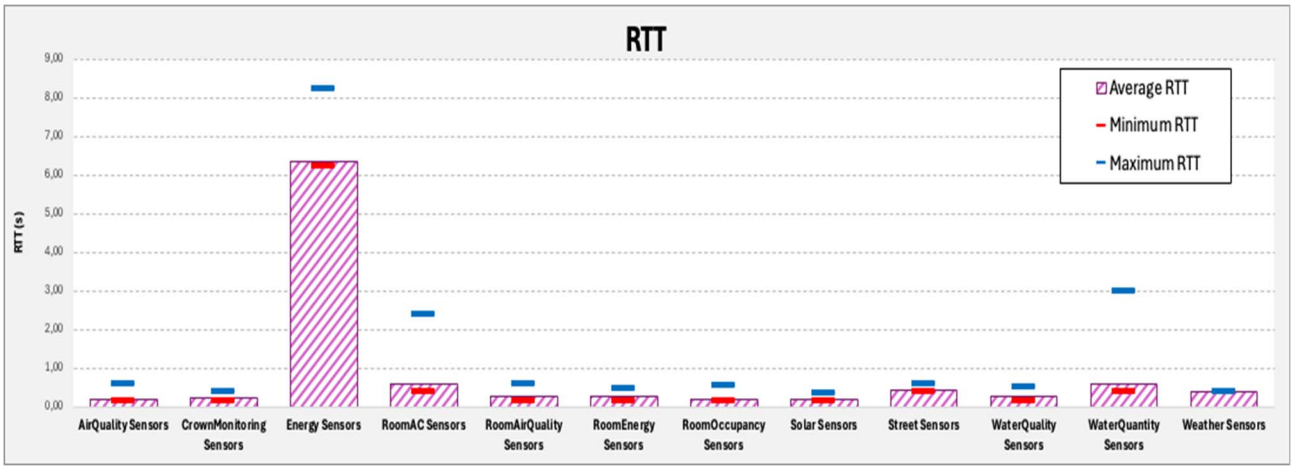

Figure 5.5: Simulated response time for the different families of sensors with distribution on the MN-CSEs

6 Use Case - eHealth Monitoring

6.0 Foreword

This use case is an instantiation of the generic event Trigger/Periodic Event IoT system use case defined in Clause 6 of ETSI TR 103 839 [i.1]. It models an application communicating to an oneM2M CSE.

From the application side, it illustrates an eHealth system for patient monitoring with chronic health disease. In such a system, an important requirement is its ability, at any time, to collect and store patient data in a periodic way, but also to react in real-time to sporadic events related to abnormal situations.

The chosen use case illustrates these different situations by considering three main types of actors namely:

- *the patient* with a chronic health disease that uses sensor devices for medical status measurements (heart rate, glycemia).
- *the physician* that follows remotely the pathology of its patient through an application that can access patient medical measures.
- *the emergency physician* or cardiologist at the hospital who is called in urgency to treat an acute episode (heart attack, diabetic coma) and needs near-real-time access to the patient constants to stabilize the patient.

The oneM2M logical infrastructure and services that will instantiate the various elements of the e-Health monitoring system is classically based on:

- a large set of bio sensors worn by patients that use an eHealth monitoring application as a gateway to connect to the devices (an ASN on a smartphone).
- A tree-based infrastructure of CSE nodes corresponding to the dispatching of the data collection to Physicians (MN-nodes) at the leaves of the tree and a root node (IN-node) hosted by the hospital cloud server.
- the health sensors use **a mix of technologies**, highlighting the heterogeneous nature of selecting the devices with the best performance with BLE, Wi-Fi® and ethernet protocol supports.

From a performance point of view, the main objective here is to highlight the different policies for collecting data from multiple (potentially numerous) sensors in an IoT system and to compare the efficiency of different services of CSE (notification, subscription, pooling) for capturing these requirements and evaluate the scalability of such a system with reference to the underlying service and hardware infrastructure. This simulation can be used to characterize the performance requirements and protocols desired by the end-to-end system deployed by a health care facility.

6.1 Representation

6.1.0 Foreword

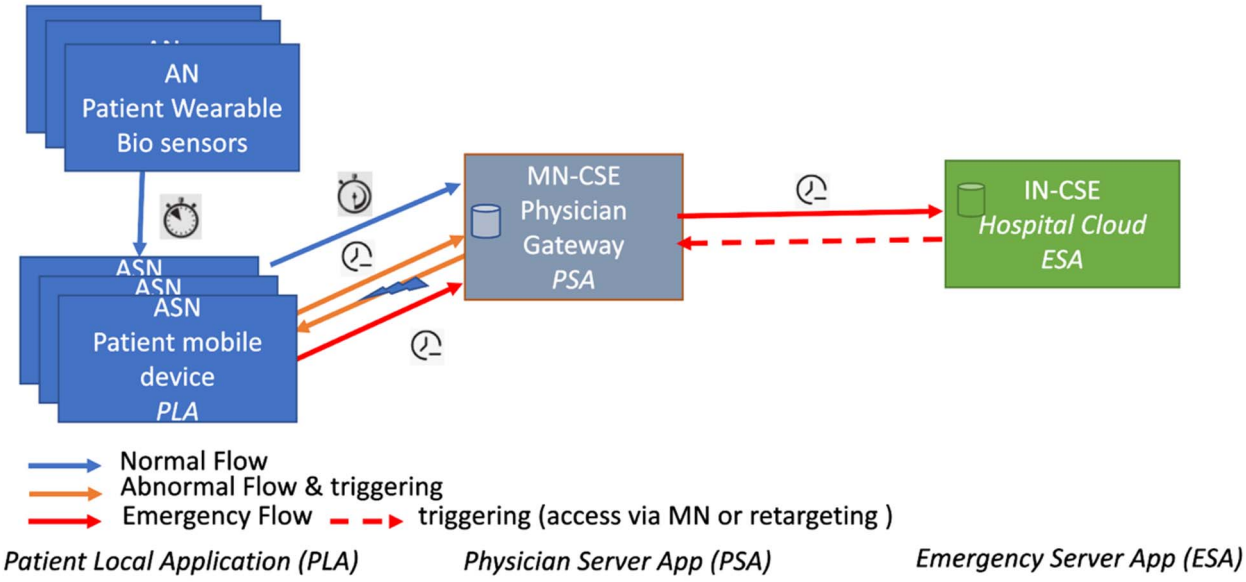

Figure 6.1: E-Health oneM2M deployed architecture

The oneM2M logical infrastructure that will instantiate the various elements of the e-Health monitoring system is shown on Figure 6.1.

In the use case three applicative situations are envisaged: The associated flows are represented on the Figure 6.1 in plain or dashed lines according to the three main situations:

- A normal flow:
  - Every X minute, sensors measure the patient biological constants and send them, when possible, to the Patient Local Application (PLA).
  - Periodically (e.g. every week) or in case of presential or remote consultation, the last data are deposited on the Physician Data Space and processed.
- An abnormal flow with triggering: In case of an abnormal evolution of data detected by the Physician Server App (PSA), an alert with the associated data is sent to the Physician.
- The Physician triggers action for a future consultation or new directives in the medical treatment of the patient.
- An emergency flow: In case a critical data threshold or value is detected by the Patient Local App, an emergency alarm is sent to the hospital emergency service. The emergency practician has real-time access to the health data constants of the patient through an Emergency Server App (ESA).

These requirements are specified in the OMNeT++ simulator *onmetpp.ini* file that configures the different parameters of the system just defined [i.7].

6.1.1 High-level view of the oneM2M Application Scenario descriptor (OASd)

From the applicative side multiple parameters can be considered for tuning the simulation such as the numerosity of the applicative elements (sensors, actuators) and their transmission rates to produce/consume data that characterize the applications' generated traffic (initially expressed through the application descriptor). In the OMNeT++ tool, the sizing and the structure are instantiated in the *Network.ned* file whereas the velocity of sensors is stated in the *omnetpp.ini* file.

The OASd part corresponding to the use case is the following:

```
//Definition of application nodes sensors, actuators
Network.eHealthSensor*.application[0].actuatorCount = 0
Network.eHealthSensor*.application[0].sensorCount = 1
Network.eHealthSensor1.application[0].name = "WearableApp1"
Network.eHealthSensor1.application[0].sensor[0].baseURL = "asn1-cse"
Network.eHealthSensor1.application[0].sensor[0].baseID = 1000000
…



//Data generation rules from sensors
Network.eHealthSensor1.application[0].sensor[0].cinGenerator =
"data:{"type":"SimulatedEvent",
   "eventDistribution":{
      "type":"Constant",
      "constant":10.0
   },
   "dataSizeDistribution":{
      "type":"Constant",
      "constant":100
   }
}
…
//Definition of remote CSE for AN
Network.eHealthSensor1.application[0].remoteCSE = "asn1-cse"
Network.eHealthSensor1.application[0].remoteCSEAddress = "10.1.1.2"
…
```

The overall file (*omnetpp.ini*) can be found in the source code referenced in Annex A.

6.1.2 High-level view of the oneM2M Solution Deployment descriptor (OSDd)

Figure 6.2 illustrates the deployment of application and service on the hardware topology of the E-health system including the different communication support.

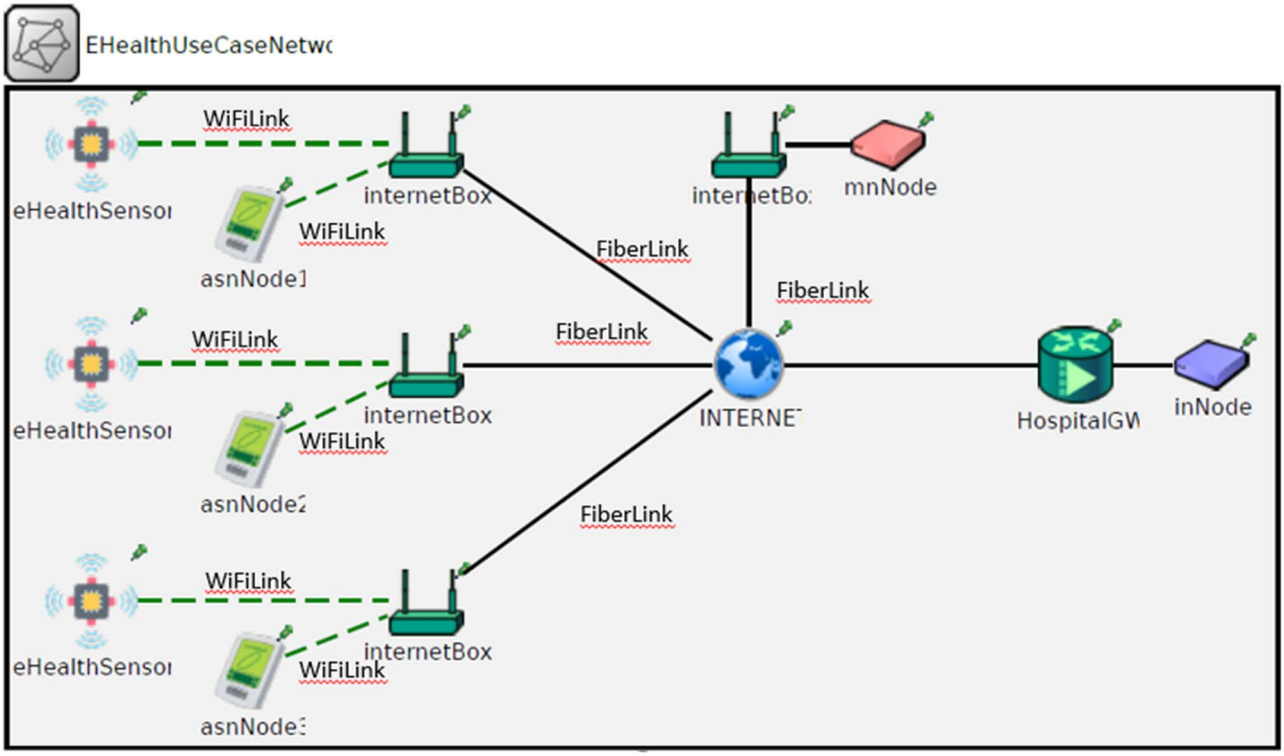

Figure 6.2: Topology view of the E-health Use Case

The sensor devices and applications appear as oneM2M logical entities such as Application Dedicated Nodes (ADNs) for the sensors, ASN-CSEs for the patient applications which collect the sensors data and carries the Patient Application Service (PLA). Middle nodes (MN-CSE) host the data collection and detection services associated with the monitoring application from the Physician side and act as a gateway able to manage multiple patients. Finally, an infrastructure IN-CSE node is the root node of the oneM2M e-Health infrastructure, able to interact with practitioners or directly with patients in case of emergency situations.

From the communication point of view, the oneM2M-based IoT solution considers medical sensor devices connected to a patient's smartphone using BLE protocol. The smartphone itself is connected to the internet through the internet gateway (i.e. Internet Box). Within the patient's home, these devices are assumed to be connected through a Wi-Fi® network (green dashed links). The practitioner has data management software connected to an oneM2M CSE (mnNode) that is connected to the Internet.

From the service deployment side, multiple parameters can be considered for tuning the simulation. The numerosity of the CSE, their interconnection and their deployment on the different hardware nodes and the variability of the protocols are interesting parameters. In the OMNeT++ tool, the sizing and the service layer structure are instantiated in the *omnetpp.ini* file and the *Network.ned* file.

Extracted from the *omnetpp.ini* file, the OSDD part contains the following parameters:

```
//Definition of the CSE infrastructure elements
Network.inNode.name = "in-cse"
Network.mnNode.name = "mn-cse"
Network.asnNode1.name = "asn1-cse"
….
//Definition of their communication properties
Network.asnNode1.name = "asn1-cse"
Network.asnNode1.nic[0].networkAddress = "10.1.1.2"
…
//Interconnection of CSEs nodes
Network.asnNode*.cse.remoteCSE = "mn-cse"
Network.asnNode*.cse.remoteCSEAddress = "10.2.0.1"
Network.mnNode.cse.remoteCSE = "in-cse"
Network.mnNode.cse.remoteCSEAddress = "10.3.0.1"
```

```
//Performances of nodes for a given oneM2M stack: processorUsage, memoryUsage, …

Network.mnNode.cse.performanceDescriptor = "data:{"PerformanceDescriptor":{"name": "m2m-sample-
stack","product": "Sample oneM2M Stack","version": "v1.0",
"minProcessorUsage": 5000,
"minMemoryUsage": 200000.0,
"resourcePerformances":
{"AE": {
         "processorUsage": {
                  "createOperation": 12766,
                  "retreiveOperation": 250,
                  "updateOperation": 845,
                  "deleteOperation": 124},
         "memoryUsage":
….
```

The NED file contains the underlying communication infrastructure.

```
connections:
   inNode.toNetwork[0] <--> FiberLink <--> HospitalGW.link++;
   internetBox.internetLink <--> FiberLink <--> INTERNET.link++;
   HospitalGW.internetLink <--> FiberLink <--> INTERNET.link++;
   internetBox1.internetLink <--> FiberLink <--> INTERNET.link++;
```

The overall files (*omnetpp.ini* and *Network.ned*) can be found in the source code referenced in Annex A.

6.1.3 High-level view of the oneM2M CSE Performance descriptor (OCPd)

Finally, performance descriptors characterize the deployment of service and communication onto the hardware resources available in the physical infrastructure.

- the available hardware resources on CSE nodes (amount of RAM in bytes & processing power in terms of instructions per second). This information is collected from the performance descriptor of the hardware node and defined in the omnetpp.ini file.
- the characteristics of the communications links (bandwidth, latency, packet/bit error rates) take also part of the omnetpp.ini file.
- the performance characteristics of the CSE nodes (expressed through the performance descriptor generated by the oneM2M stack profiler).

Here again, this information is defined in the omnetpp.ini file as illustrated below:

- The average processor usage, the RAM memory occupancy and the disk usage for the different CRUD operations generated in the system.

```
// Hardware nodes characteristics, RAM instruction Per Second
Network.eHealthSensor*.ramManager.maximumBytes = 200000
Network.eHealthSensor*.cpuManager.IPS = 1000

Network.asnNode1.cpuManager.IPS = 2548
Network.asnNode2.cpuManager.IPS = 2550
Network.asnNode3.cpuManager.IPS = 2552
Network.asnNode1.ramManager.maximumBytes = 50000
Network.asnNode2.ramManager.maximumBytes = 50000
Network.asnNode3.ramManager.maximumBytes = 50000

Network.mnNode.cpuManager.IPS = 5000
Network.mnNode.ramManager.maximumBytes = 200000

Network.inNode.cpuManager.IPS = 5000
Network.inNode.ramManager.maximumBytes = 200000
```

The effective RAM and CPU and disk available in the different computing hardware.

```
**.ramManager.maximumBytes = 200000
**.cpuManager.IPS = 20000
```

The overall file (*omnetpp.ini*) can be found in the source code referenced in Annex A.

6.2 Results

From a performance point of view, the main objective here is to highlight the different policies for collecting data from multiple (potentially numerous) sensors in an IoT system and to compare the efficiency of different services of CSE (notification, subscription, pooling) for capturing these requirements and evaluate the scalability of such a system with reference to the underlying service and hardware infrastructure.

The related KPIs that are computed and analysed are:

- the identification of bottlenecks in certain communications by analysing for each CSE the message queue state;
- Under-sampling issues of the device nodes;
- Real-time data recording capacity;
- Stressing the system: reaction time and registration time are checked in the context of a stressed system i.e. by increasing the ratio device/gateway and/or gateway/server;
- Concerning the deployed architecture, the number of end devices is parametrized as well as the number of gateways and servers.

Thanks to the instrumented code placed inside the simulation modules, multiple KPIs are measured during the simulation.

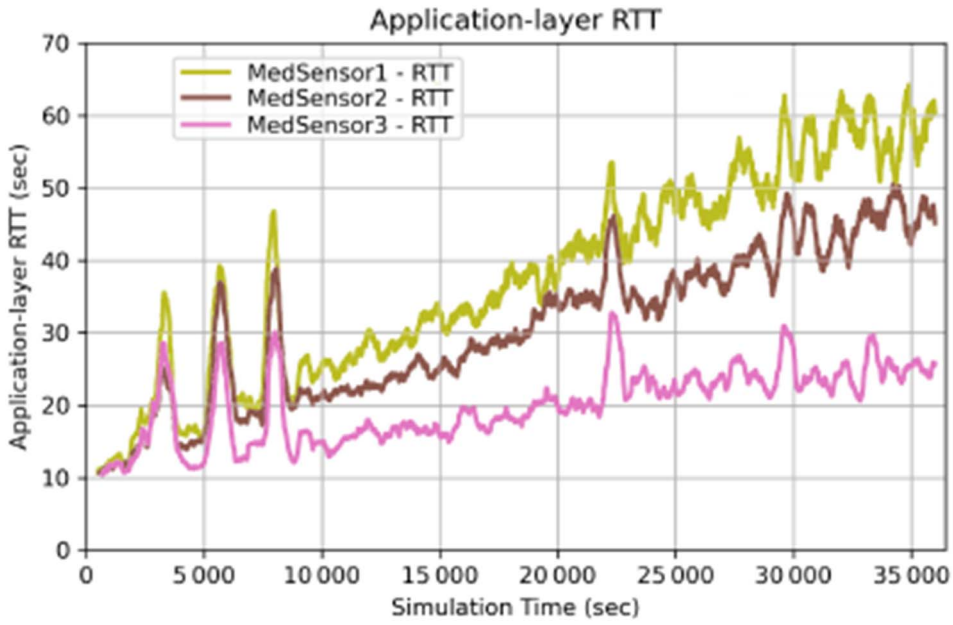

Figure 6.3: Application-Layer RTT

Figure 6.3 draws the results of the normal flow simulation and its associated RTT. In this demonstration simulation, ASN nodes are given different processing capabilities (asnNode3 > asnNode2 > asnoNode1) but IoT sensors attached to these nodes generate similar traffic (periodic messages for the normal flow and exponential traffic for the alternative flow). Also, the IN Node is deliberately under-provisioned in terms of CPU and RAM given the overall traffic it manages.

Due to the different resources available at the ASN nodes, the application layer RTT grows during the simulation. The random peaks correspond to the messages that are sent to the IN nodes (alternative data flow) where this node is always saturated due to its poor resources. These KPI include information such as:

1) application-layer RTT representing time difference between message sending time and acknowledgement reception;

2) the message processing time in a CSE; and

3) the message queue occupancy in a CSE.

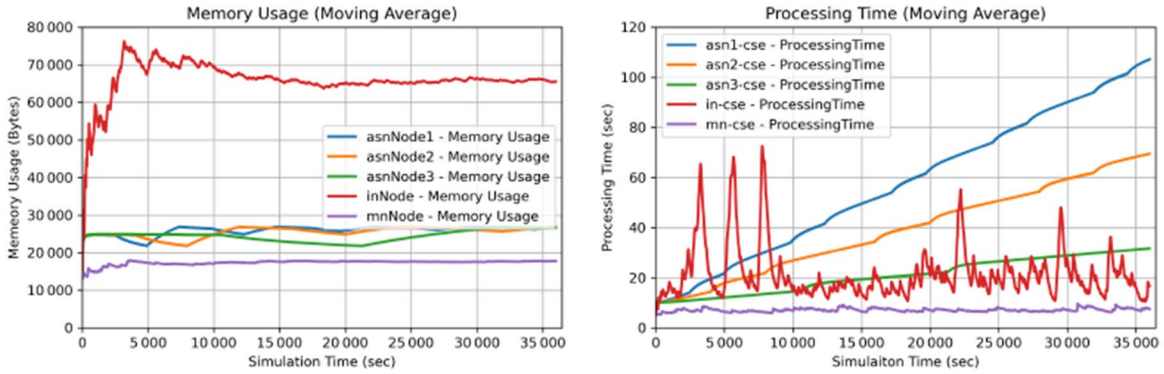

Figure 6.4: CSE Memory usage and Processing Time

Figure 6.4 shows examples of these two KPIs generated at the end of the simulation. In the left part the (simulated) RAM occupancy grows on ASN Node as messages are queued before being processed by the CSE (as it happens in real CSEs). The right part shows the impact of this phenomenon on the processing time since it is the sum of the waiting time in the queue and the actual processing time (provided in the performance descriptor of the CSE). This figure also shows how the memory usage and the processing time of messages in the case of the IN Node are always high. It is also worth noticing that the MN node is behaving well since the node is well provisioned. These experiments clearly show that the resources allocated to the IN Node are undersized to handle the overall traffic. Therefore, our simulator can be used in order to seek the best deployment plan by tuning the configuration parameters (in particular nodes resources).

7 Use Case - Traffic Light Control and Monitoring

7.0 Foreword

Based on the use case described in clause 7 of [i.1], the demonstrator traffic lights illustrates the scalability capacity in a oneM2M architecture and the impact of oneM2M communication features to cope with the temporal and synchronization requirements of the traffic light system.

One aspect of this synchronization problem is the inability to monitor and update the schedule of the traffic lights at these intersections. This represents a good use case for an IoT solution and a sub-component of a larger smart city deployment (Figure 7.1). This scenario illustrates the capability of the simulator to mix several domains and technologies and its ability to get results that could be compared to real execution on real host and oneM2M stack.

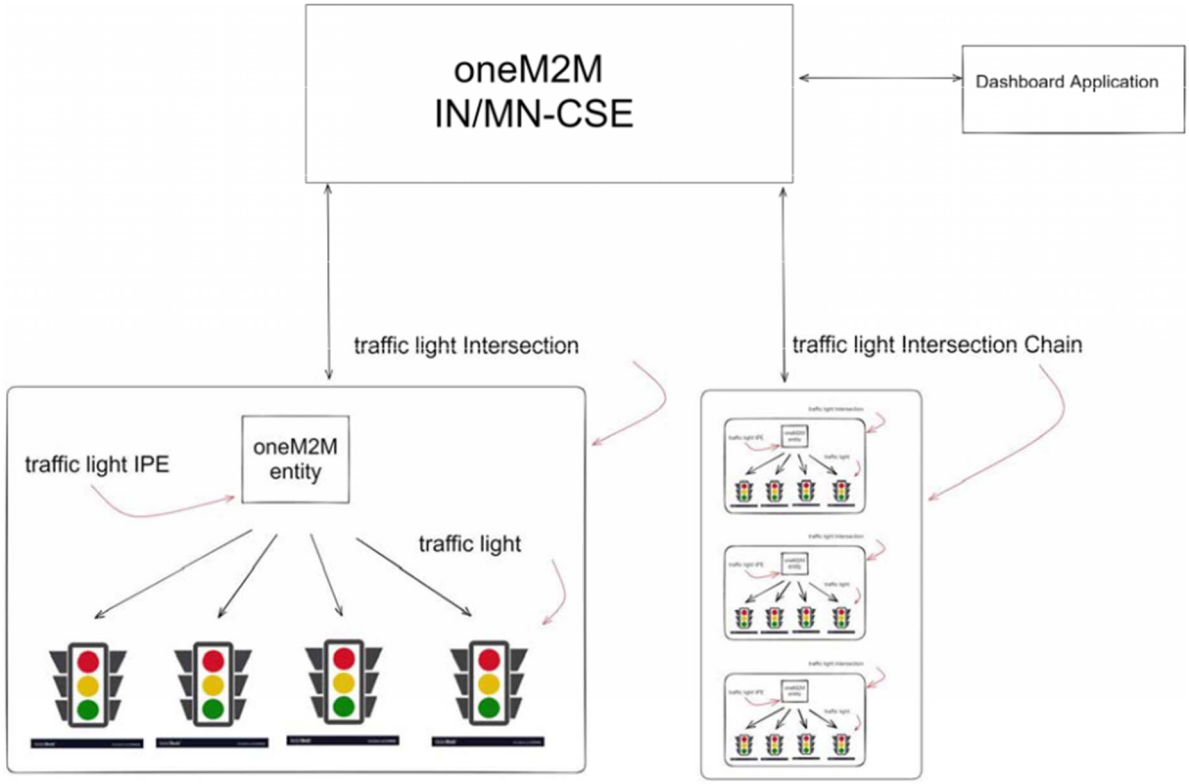

Figure 7.1: Architecture of traffic light system

7.1 Representation

7.1.1 High level view of oneM2M Application Scenario descriptor (OASd)

OASd meta-model has been described in [i.2]. To model this scenario and map to the capabilities of the simulator, several CSE have been created. Each category of sensors has been described in terms of numbers, law to express time between two send of data and the size of message. The links between sensors and CSE have been expressed (Figure 7.2).

The OASd part corresponding to the use case is the following:

```
//Definition of application nodes sensors, actuators
Network.intersection*.application[0].actuatorCount = 3
Network.intersection*.application[0].sensorCount = 3
Network.intersection*.application[0].sensor[0].baseURL = "in-cse"
Network.dashBoard.application[0].actuator[0].baseURL = "in-cse"

…

//Data generation rules from sensor

Network.intersection*.application[0].sensor[0].cinGenerator =
"data:{"type":"SimulatedEvent",
          "eventDistribution":
          {
                    "type":"Constant",
                    "constant":10.0
          },
          "dataSizeDistribution":
          {
                    "type":"Constant",
                    "constant":100
          }}"
```

```
…
//Data collection from Actuators (Polling or SubNotify strategy
[Config Polling]

Network.dashBoard.application[0].actuator[0].isSubscribing = false
Network.dashBoard.application[0].actuator[0].isPolling = true
Network.dashBoard.application[0].actuator[0].pollingDelay = ${10, 20, 30}

[Config SubNotify]
Network.dashBoard.application[0].actuator[0].isSubscribing = true
Network.dashBoard.application[0].actuator[0].isPolling = false
…
Network.dashBoard.application[0].actuator[0].pollingTarget = "AE_StreetLights1-1
AE_StreetLights1-2 AE_StreetLights1-3 AE_StreetLights1-4 AE_StreetLights1-5 AE_StreetLights2-1
AE_StreetLights2-2 AE_StreetLights2-3 AE_StreetLights2-4 AE_StreetLights2-5 AE_StreetLights3-1
AE_StreetLights3-2 AE_StreetLights3-3 AE_StreetLights3-4 AE_StreetLights3-5"
…
```

The overall file (*omnetpp.ini)* can be found in the source code referenced in Annex A.

7.1.2 High-level view of the oneM2M Solution Deployment descriptor (OSDd)

Figure. 7.2 illustrates the deployment of application and service on the hardware topology of the traffic lights system including the different communication supports.

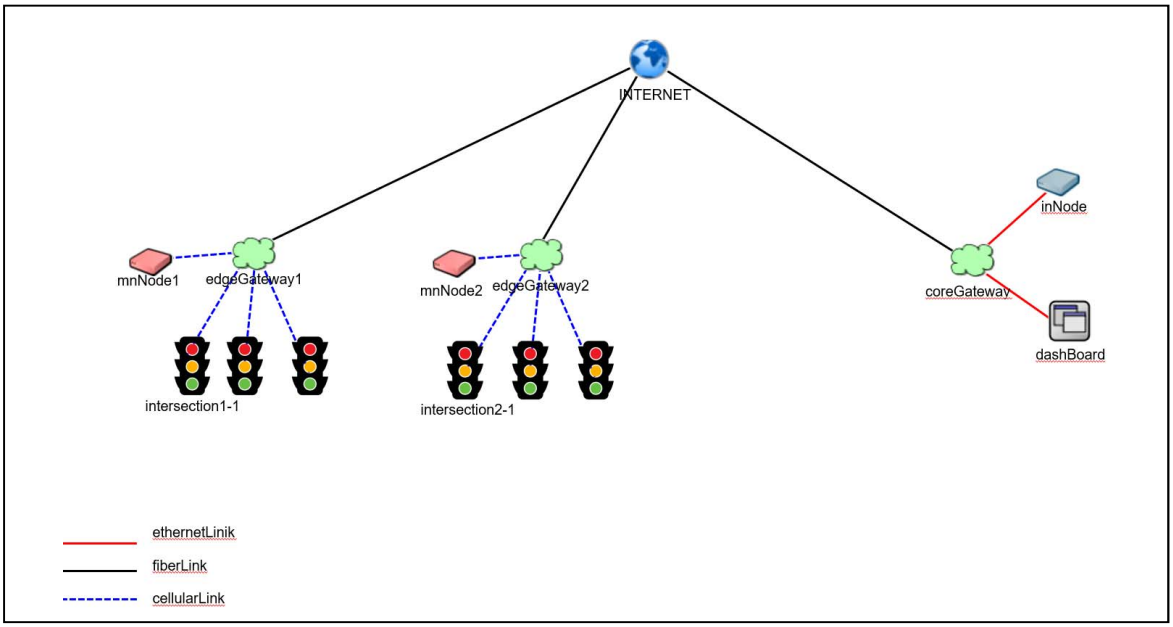

Figure 7.2: Topology view of the Traffic light Use Case

The different sensors devices and actuators are represented by traffic lights, These devices are managed (collection of data and actuations) by MN-CSE middle nodes. An infrastructure IN-CSE node is the root node of the oneM2M traffic lights infrastructure. An application entity node is connected to IN as a dashboard of the overall system.

From the communication point of view, the oneM2M-based IoT solution considers cellular links between AE nodes. Fiber links connect the middle nodes to the infrastructure node.

From the service deployment side, multiple parameters can be considered for tuning the simulation such as the numerosity of the MN-CSE and their interconnection with IN node but also the policy for collecting and sending data to the actuators (polling service of sub-notify service).

The structure is instantiated in the *Network.ned* file whereas velocity and the policy for actuators are stated in the *omnetpp.ini* file.

Extracted from the omnetpp.ini file, the OSDD part contains the following parameters:

```
//Definition of the CSE infrastructure elements
inNode: CSENode {
    name = "in-cse";
    nic[0].name = "eth0";
    nic[0].networkAddress = "10.0.0.1";
}
mnNode[cseCount]: CSENode {
    name = "mn" + string(index+1) + "-cse";
    cse.remoteCSE = "in-cse";
    cse.remoteCSEAddress = "10.0.0.1";
    nic[0].name = "eth0";
    nic[0].networkAddress = "10.0." + string(index+1) + ".1";
}
…
//Performances of nodes for a given oneM2M stack: processorUsage, memoryUsage, …
**.cse.performanceDescriptor = "file://../../descriptors/cse/om2m_v1.1.json"
**.localDB.performanceDescriptor = "file://../../descriptors/db/sample-db_v1.0.json"
Network.intersection*.application[0].sensor[0].baseURL = "in-cse"
Network.dashBoard.application[0].actuator[0].baseURL = "in-cse"

…
```

The NED file contains the underlying communication infrastructure connections:

```
connections allowunconnected:
  coreGateway.internetLink <--> FiberLink <--> INTERNET.link++;
  inNode.toNetwork[0] <--> EthernetLink <--> coreGateway.link++;
  dashBoard.toNetwork[0] <--> EthernetLink <--> coreGateway.link++;

  for i=0..cseCount-1 {
     mnNode[i].toNetwork[0] <--> CellularLink <--> edgeGateway[i].link++;
     edgeGateway[i].internetLink <--> FiberLink <--> INTERNET.link++;
  }
  for j=0..deviceCount-1 {
    intersection[j].toNetwork[0] <--> CellularLink <--> edgeGateway[int(j/devicesPerCSE)].link++;
  }
```

The overall files (*omnetpp.ini* and *Network.ned*) can be found in the source code referenced in Annex A.

7.1.3 High-level view of the oneM2M CSE Performance descriptor (OCPd)

Finally, performance descriptors characterize the deployment of service and communication onto the hardware resources available in the physical infrastructure:

- the available hardware resources on CSE nodes (amount of RAM in bytes & processing power in terms of instructions per second). This information is collected from the performance descriptor of the hardware node and defined in the omnetpp.ini file;
- the characteristics of the communications links (bandwidth, latency, packet/ bit error rates) take also part of the omnetpp.ini file;
- the performance characteristics of the CSE nodes (expressed through the performance descriptor generated by the oneM2M stack profiler).

Here again, this information is defined in the omnetpp.ini file as illustrated bellow:

- The average processor usage, the RAM memory occupancy and the disk usage for the different CRUD operations generated in the system including the retargeting.

```
// Hardware nodes characteristics, RAM instruction Per Second
**.ramManager.maximumBytes = 200000
**.cpuManager.IPS = 40000
```

The overall file (*omnetpp.ini*) can be found in the source code referenced in Annex A.

7.2 Results

In those simulations, two phenomena are studied. The first one is to evaluate the impact of polling time in a polling strategy request. The second one is to see the impact of changing the polling strategy with a subscribe notification strategy. Two KPIs are measured: the processing time needed and the memory usage in the CSE.

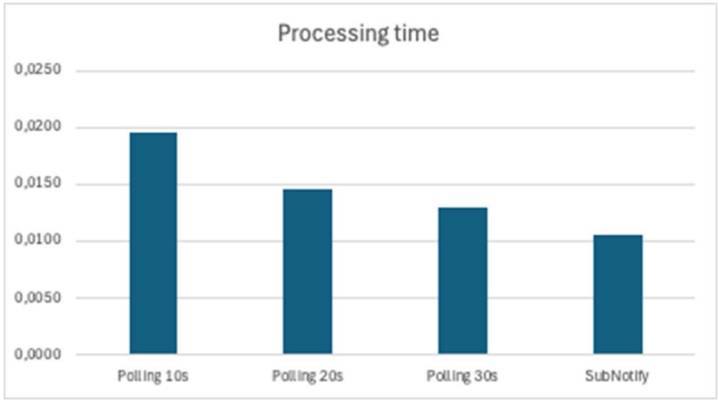

Figure 7.3: Processing time for traffic light Use Case

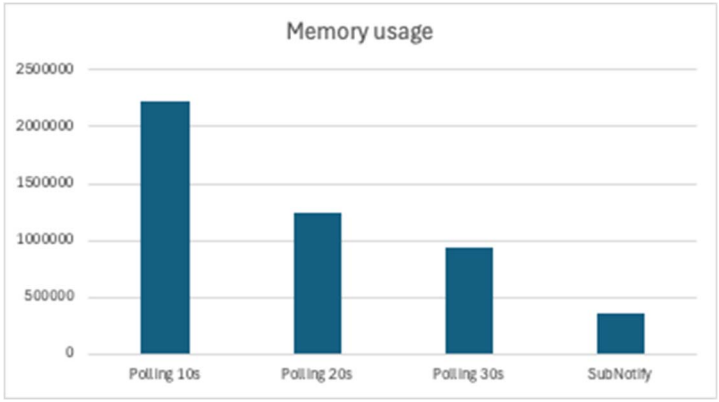

Figure 7.4: Memory usage for traffic light Use Case

The curves confirm that the polling time has a large impact of processing time (Figure 7.3) with an average processing time of 0,0195 s per request for a polling every 10 s that decrease to 0,0131 s per request for a polling every 20 s. It also shows the impact in memory (Figure 7.4). This shows the impact of queuing the request from the network and the possible saturation of sequential processor.

Finally, the subscription/notification mechanism is more interesting even if the polling time is only 30 s.

8 Conclusions

The present document presents three demonstrators which, each of them, highlighted the simulation capabilities of the tools developed in the TTF and described in ETSI TR 103 841 [i.3]. The present document provides the relevant configuration elements of these three demonstrators, based on the use cases described in [i.1]. They highlight classical behaviours in IoT system deployment by illustrating features of the simulation tools in terms of KPI analysis.

More specifically :

- The Smart Campus use case demonstrates the simulator's ability to express IoT systems with many IoT nodes. It also demonstrates the simulator's ability to express heterogeneous systems in terms of communication networks, CSE implementation, as well as taking into account advanced oneM2M parameterization related to the choice of data storage on the oneM2M nodes (distributed storage on the MN nodes, or centralized storage on the IN node).

- The eHealth use case demonstrates the simulator's ability to describe scenarios where the IoT application exhibits multiple data flows to different destinations (ASN node, MN node, IN node) without having to describe the application logic.

- The Traffic Lights use case demonstrates the simulator's ability to simulate advanced oneM2M features, namely the Polling and Subscription/Notification mechanisms. It also demonstrates the simulator's ability to conduct parametric studies by facilitating the description of IoT systems by defining a parameterizable topology, thus making it easy and quick to have variants of the simulated system (for example, by varying the number of MN or IoT nodes, or the polling frequency).

Annex A:
Source code

See https://labs.etsi.org/rep/iot/smartm2m-onem2m-performance-evaluation/onem2m-simulator.

NOTE: See [i.8].

Annex B:
YAML Schemas for OASd and OCPd

```yaml
OASD:
  $schema: 'http://json-schema.org/draft-04/schema#'
  description:
    oneM2M Application Scenario descriptor (OASd). The OASd is a JSON document is
    used to describe message generation within an application in terms of timing
    and data size.
  type: object
  properties:
    type:
      type: string
      description:
        The type of the event generator. It can be either SimulatedEvent or RecordedEvent.
      enum:
        - SimulatedEvent
        - RecordedEvent
    eventDistribution:
      type: object
      description:
        The distribution of the time between events.
        If the type is Constant, the constant field is used.
        If the type is Exponential, the lambda field is used.
        If the type is Uniform, the lowerBound and upperBound fields are used.
      properties:
        type:
          type: string
          description:
            The type of the distribution. It can be either Constant, Exponential or Uniform.
          enum:
            - Constant
            - Exponential
            - Uniform
        constant:
          type: number
          description: Time period between two consecutive events.
        lambda:
          type: number
          description: The mean of the exponential distribution.
        lowerBound:
          type: number
          description: The minimum value of the uniform distribution.
        upperBound:
          type: number
          description: The maximum value of the uniform distribution.
    dataSizeDistribution:
      type: object
      description:
        The distribution of the size of the data message (oneM2M Content Instance).
        If the type is Constant, the constant field is used.
        If the type is Uniform, the lowerBound and upperBound fields are used.
      properties:
        type:
          type: string
          description: The type of the distribution. It can be either Constant or Uniform.
          enum:
            - Constant
            - Uniform
        constant:
          type: number
          description: The constant value of the data size.
        lowerBound:
          type: number
          description: The minimum value of the uniform distribution.
```

```
        upperBound:
          type: number
          description: The maximum value of the uniform distribution.
  required:
    - type
    - eventDistribution
    - dataSizeDistribution
```

```
OCPD:
  $schema: 'http://json-schema.org/draft-04/schema#'
  description:
    oneM2M CSE Performances descriptor (OCPd). The OCPd is a JSON document that
    used to describe the performances of an oneM2M CSE in terms or hardware resources'
    usage (CPU, Memory, and Disk). These characteristics are per resource type and
    per operation type (Create, Retrieve, Update or Delete).
    operation.
  type: object
  properties:
    name:
      type: string
      description: Name of the performance descriptor of an oneM2M CSE
    product:
      type: string
      description: Name of the product for which the performance descriptor is defined
    version:
      type: string
      description: Version of the product for which the performance descriptor is defined
    minProcessorUsage:
      type: number
      description: Minimum processor usage for the product (in instructions per second)
    minMemoryUsage:
      type: number
      description: Minimum memory usage for the product (in bytes)
    resourcePerformances:
      type: object
      description: Performance indicators for the supported resources
      properties:
        AE:
          type: object
          description: Performance indicators for the Application Entity (AE) resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
        CNT:
          type: object
          description: Performance indicators for the Container (CNT) resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
        CIN:
          type: object
          description: Performance indicators for the Content Instance (CIN) resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
        LASTEST:
          type: object
          description: Performance indicators for the latest virtual resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
        ACP:
```

```
          type: object
          description: Performance indicators for the Access Control Policy (ACP) resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
        SUB:
          type: object
          description: Performance indicators for the Subscription (SUB) resource
          properties:
            performance:
              type: object
              $ref: '#/definitions/ResourcePerformance'
      required:
        - AE
        - CNT
        - CIN
        - LATEST
        - ACP
        - SUB
  required:
    - name
    - product
    - version
    - minProcessorUsage
    - minMemoryUsage
    - resourcePerformances
definitions:
  ResourcePerformance:
    type: object
    properties:
      processorUsage:
        type: object
        $ref: '#/definitions/CRUDPerformance'
      memoryUsage:
        type: object
        $ref: '#/definitions/CRUDPerformance'
      diskUsage:
        type: object
        $ref: '#/definitions/CRUDPerformance'
    required:
      - processorUsage
      - memoryUsage
      - diskUsage
  CRUDPerformance:
    type: object
    properties:
      createOperation:
        type: number
      retreiveOperation:
        type: number
      updateOperation:
        type: number
      deleteOperation:
        type: number
    required:
      - createOperation
      - retreiveOperation
      - updateOperation
      - deleteOperation
```

Annex C: Bibliography

- oneM2M TS-0008 (V3.9.0): "CoAP Protocol Binding".
- oneM2M TS-0009 (V3.9.0): "HTTP Protocol Binding".
- ETSI TS 118 110 (V3.1.0): "oneM2M; MQTT Protocol Binding (oneM2M TS-0010 version 3.1.0 Release 3)".

History

Document history		
V1.1.1	February 2025	Publication