FACULTY OF ENGINEERING
AND ARCHITECTURE

Developing Software Services in Smart Cities based on Edge to Cloud Orchestration

Jaro Robberechts
Student number: 01606256

Supervisor: Prof. dr. Bruno Volckaert
Counsellor: Tom Goethals

Master's dissertation submitted in order to obtain the academic degree of
Master of Science in Information Engineering Technology

Academic year 2019-2020

GHENT
UNIVERSITY

Norwegian University of Science and Technology
Faculty of Computer Science
Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

Supervisors:	Prof. Dr. Sobah Abbas Petersen
	Prof. Dr. Bruno Volckaert
Co-supervisor:	Dr. Amir Sinaeepourfard
Counsellor:	Tom Goethals

Norwegian University of Science and Technology
Faculty of Computer Science

Ghent University
Faculty of Engineering and Architecture
Department of Information Technology

This Master's dissertation has been established in collaboration with the ZEN research center.

Master's dissertation submitted in order to obtain the
academic degree of
Master of Science in Information Engineering Technology
Academic Year 2019-2020

Acknowledgment

Throughout the creation of this Master's dissertation, I have received a great deal of support.

First and foremost, I would like to thank my supervisor from the NTNU, prof. dr. Sobah Abbas Petersen, for giving me the opportunity to write this Master's Thesis at the NTNU and in collaboration with the ZEN Research Center. Secondly, I want to thank my supervisor from the UGent, prof. dr. Bruno Volckaert, for always giving me good advice and support when I needed it the most. This really kept me motivated. Next, I want to express my gratitude towards dr. Amir Sinaeepourfard, for supporting me and pushing my limits throughout this semester. You thought me many things that will stay with me for the rest of my life. You also gave me the opportunity of publishing my first paper. I am thankful for this tense but exciting experience and it was a pleasure to be working with you. I also want to thank Tom Goethals for his support, feedback, and motivation.

Furthermore, I want to thank Wouter Vandendriessche for our endless talks about each others work in Moholt. I also want to thank my family and friends for always being there when I needed them. Especially, Valerie, even though we were separated during my five-month stay in Norway, you always kept me motivated and happy.

Ghent, August 2020
Jaro Robberechts

Permission for use of content

Ghent, August 2020
Jaro Robberechts

Table of Contents

List of Figures

List of Tables

Acronyms

AI Artificial Intelligence. 20, 34

API Application Programming Interface. 30, 32, 35

BS Base Station. 14, 20, 45

CaaS Cloud-as-a-Service. 6, 13

caaS cloudlet-as-a-Service. 13

CC Cloud Computing. 1, 6, 8, 11, 28, 31, 54

CPU Central Processing Unit. ix, 52, 53, 55–57

D2C Distributed-to-Centralized. xvii, 24

DC2C Decentralized-to-Centralized. xvii

DSM Demand-Side Management. 39

DSU Dynamic Software Update. 48

E2CaaS Edge-to-Cloud-as-a-Service. ix, 13, 25, 40, 41, 43, 45, 49, 54, 58–60

EaaS Edge-as-a-Service. 14

EC Edge Computing. 1, 9–11, 19, 25, 27, 28, 32, 33, 45

EMS Energy Management System. ix, xvii, xix, 2, 3, 37–40, 42–44, 48, 49, 51, 52, 59

EU European Union. 20, 44

FaaS Fog-as-a-Service. 13

FC Fog Computing. 1, 8–11, 16, 32, 45

GDPR General Data Protection Regulation. 20, 44

GPU Graphics Processing Unit. 34

I2CM-IoT Integrated and Intelligent Control and Monitoring of IoT. 14, 46, 48, 54

IaaS Infrastructure-as-a-Service. ix, 6, 7

ICT Information and Communication Technology. xvii, xix, 2, 8, 12, 16, 22–24, 26–28, 35, 44, 49, 60

IoT Internet of Things. ix, xvii, xix, 1, 6, 8, 9, 14, 20, 22, 25, 27–29, 31, 33–35, 42, 52–57

IT Information Technology. xvii

KPI Key Performance Indicator. 16, 35, 46, 48

MEC Mobile Edge Computing. 10, 11, 20, 26, 27, 29, 32, 33, 42

MECaaS MEC-as-a-Service. 14

MNO Mobile Network Operator. 11

MQTT Message Queuing Telemetry Transport. 21, 33, 34, 48, 50, 58, 60

NFV Network Function Virtualization. 27, 32, 33

OS Operating System. 30, 53

PaaS Platform-as-a-Service. ix, 6, 7, 33

RAN Radio Access Network. 10, 11, 14, 28

SaaS Software-as-a-Service. ix, 6, 7

SDK Software Development Kit. 35

SDN Software-Defined Networking. ix, 27, 31, 32, 47, 48

SP Service Provider. 6, 9, 11, 28, 32, 33

SSH Secure Shell. 19

VM Virtual Machine. ix, 29, 30, 33

VNF Virtual Network Function. 33

ZEN Zero Emission Neighborhoods. i, ix, xvii, xix, 2, 3, 37, 38, 40, 41, 43–49, 59, 60

Abstract

Complex challenges such as fast population growth, pollution, safety, and climate chance urge the need for newer and better Information Technology (IT) services. Smart Cities are one of the scenarios where these services are utterly important. The main goal of Smart Cities is to enhance the Quality of Life of its inhabitants by providing services that can tackle the previously mentioned challenges. Smart Cities can leverage their wide variety of Information and Communication Technology (ICT) components that are built around Internet of Things (IoT) to offer these Software Services. However, managing all these ICT components is challenging. Consequently, the demand for ICT architectures in Smart Cities is high. Traditional solutions are based on centralized ICT architectures using *Cloud*-based technologies. With services shifting towards the edge of the network and the *Cloud* coming up short on many levels, novel distributed architectures are gaining popularity. Many solutions that can manage the ICT components from the edge of the network to the *Cloud* through distributed technologies, such as Distributed-to-Centralized (D2C) and Decentralized-to-Centralized (DC2C) architectures, have already been proposed. This thesis aims to give a general overview of the various technologies and methods related to large-scale Software Services in Smart Cities using different multilevel technologies. Some research questions are identified to provide the reader with a clear objective throughout this work. Based on a thorough literature review, a model on how to develop Software Services in Smart Cities is presented. Afterward, the knowledge gained from this literature study is applied onto the ZEN Research Center. This second part of the thesis explores the possibilities of EMS Software Services in Smart Neighborhoods as a use-case. The final part of this thesis gives an example of a simplified Software Service that can be used for EMSs and compares a couple scenarios to visualise the impact of using a distributed layout.

Nederlands Abstract
Dutch Abstract

Recent, door de opkomst van uitdagingen zoals onder meer een snelle demografische groei, privacy issues, en klimaatsverandering, wordt de vraag naar "slimmere" services en applicaties steeds groter. Vooral in steden, waar veel mensen dicht bij elkaar wonen en werken, spelen dit soort services nu al een grote rol. Zo een "slimme" stad of Smart Cities is gebouwd rond ICT componenten en IoT toestellen. Deze componenten en toestellen kunnen gebruikt worden om data op te halen, die dan op zijn beurt kan dienen om services aan te bieden voor de inwoners. Het blijft echter een moeilijke kwestie om al deze toestellen te beheren. Traditioneel wordt dit gedaan aan de hand van gecentraliseerde technologieën die zich in de Cloud bevinden. Tegenwoordig is de vraag naar services echter aan het verschuiven richting de rand van het netwerk. Als reactie hierop zijn er al heel wat gedistribueerde oplossing voorgesteld. Dit onderzoek tracht een duidelijk overzicht te bieden van de verscheidene technologieën en methodes die kunnen bijdragen tot het ontwikkelen van Software Services, van de rand van het netwerk tot aan de Cloud. Gebaseerd op een grondig literatuur onderzoek stelt deze studie een architectuur voor die de meeste voorkomende gedistribueerde en gecentraliseerde technologieën in een Smart City plaats. Daarnaast wordt een stappenplan voor het ontwikkelen van Software Services uitgewerkt. In het tweede deel van dit werk, worden deze modellen toegepast op het ZEN Research Center in Noorwegen. De mogelijkheden voor het bouwen van EMS Software Services in Smart Cities en Smart Neighborhoods worden hierbij onderzocht. In het laatste hoofdstuk wordt een voorbeeld van een versimpelde Software Service geïmplementeerd door gebruik te maken van containerization. Enkele scenario's worden vergeleken om de impact van een gedistribueerd platform te visualiseren.

Developing Software Services in Smart Cities based on Edge to Cloud Orchestration

Jaro Robberechts

Supervisor(s): Sobah Abbas Petersen, Bruno Volckaert, Amir Sinaeepourfard, and Tom Goethals

Abstract—**Complex challenges such as fast population growth, pollution, safety, and climate chance urge the need for newer and better Information Technology (IT) services. Smart Cities are one of the scenarios where these services are utterly important. The main goal of Smart Cities is to enhance the Quality of Life of its inhabitants by providing services that can tackle the previously mentioned challenges. Smart Cities can leverage their wide variety of Information and Communication Technology (ICT) components that are built around Internet of Things (IoT) to offer these Software Services. However, managing all these ICT components is challenging. Consequently, the demand for ICT architectures in Smart Cities is high. Traditional solutions are based on centralized ICT architectures using *Cloud*-based technologies. With services shifting towards the edge of the network and the *Cloud* coming up short on many levels, novel distributed architectures are gaining popularity. Many solutions that can manage the ICT components from the edge of the network to the *Cloud* through distributed technologies, such as Distributed-to-Centralized (D2C) and Decentralized-to-Centralized (DC2C) technologies, have already been proposed. This study aims to give a clear overview of the various technologies and methods related to large-scale Software Services in Smart Cities using different multilevel technologies. A general overview of how to develop Software Services in Smart Cities is presented based on a thorough literature review. The knowledge gained from this broad literature study is then applied onto the ZEN Research Center. This part of the study explores the possibilities of Energy Management System (EMS) Software Services in Smart Neighborhoods, as a use-case in Smart Cities. Finally, an example of a simplified Software Service that can be used for EMSs is presented. This service is implemented in a couple of scenarios to visualize the impact of using a distributed layout.**

Keywords—**Smart City, IoT, Edge-to-Cloud, Distributed-to-Centralized ICT, Edge-to-Cloud-as-a-Service**

I. Introduction

RECENT challenges related to population growth, pollution, safety, and climate change in cities have led to the adoption of new technologies such as IoT. IoT devices constitute to the main building block that forms a Smart City. To deal with some of these challenges, and to improve the Quality of Life of the citizens, the city needs Software Services that can interact with the IoT devices on a large-scale. Traditionally, these services are located on a centralized Cloud, far away from the citizens. Due to recent demands, such as low latency and privacy, the computing power is shifting closer towards the edge of the network. Consequently, novel paradigms such as Edge Computing (EC) and Fog Computing (FC) are arising. With all the new technologies, devices, and complexities, the demand for a general unified architecture that can utilize both the benefits of the centralized Cloud-based as the distributed Edge and Fog-based technologies is high.

This study tries to fill this gap by firstly providing a general overview of the current technologies and trends for building Software Services in Smart Cities. Next, based on a literature review, a novel Edge-to-Cloud-as-a-Service (E2CaaS) architecture and a model of how to develop Software Services in Smart Cities is presented. This model consists of four main steps that guide the reader through the process of developing a Software Service. Afterward, this general model and architecture are applied onto the ZEN Research Center, resulting in an architecture and model for developing EMS Software Services in Smart Neighborhoods. Finally, a simplified Software Service that can be used for EMSs is implemented in four different scenarios. The main focus in terms of technologies when developing this service lies on containerization. Comparing the measurements of the different scenarios shows that moving the computing power and the services (partially) closer towards the citizens can reduce the amount of traffic towards and the load on the Cloud.

II. Software Services in Smart Cities

A. Background

This section briefly explains some of the most vital concepts related to Software Services in Smart Cities.

A.1 Cloud Computing

CC is a computing model where users can utilize computing resources on-demand. These resources are often managed by a Service Provider (SP). The client can save a lot of money by simply renting computing power instead of investing in expensive dedicated hardware. The four most common methods for providing Cloud-as-a-Service (CaaS) are: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), Software-as-a-Service (SaaS), and Serverless Computing.

IaaS offers virtualization, storage, and processing. This leaves a lot of freedom to the client as they can choose their own Operating System (OS) and software.

PaaS SPs allows their clients to only control the applications that are running on their servers. They cannot choose the underlying infrastructure, OS, or development tools [1]. PaaS is mostly used as a development framework as it simplifies and speeds up the application development process.

SaaS offers the client the least amount of freedom. The client rents a service and has no control over the infrastructure, platform, and application. The advantage of SaaS is that it enables the client to be quickly up and running with the least amount of effort and minimal upfront cost.

Finally, Serverless Computing offers less flexibility than PaaS because the client cannot create the whole application. The client is however able to develop single functions of the application, resulting in more freedom than with SaaS.

A.2 cloudlet Computing

cloudlets are a concept that places the computing power closer towards the end-users and datasources. A cloudlet can be a small datacenter or group of computing devices, often referred to as a ”datacenter in a box” [2].

By moving the computing power closer towards the edge of the network, the citizens can receive services with lower latency. Plus, it is easier to keep their data local and private. Finally, by distributing the computational tasks, traffic towards the Cloud can be decreased.

A.3 Fog Computing

FC is an extension of CC and, as with cloudlets, tries to move the computational power closer towards the edge of the network. Compared to cloudlets, Fog nodes have relatively low computing power and are thus mostly used for simple processing tasks [3].

A.4 Edge Computing

EC is similar to both cloudlet Computing and FC. However, EC uses devices located at the edge of the network with often more computing power than Fog nodes. Edge nodes can even offer services while being disconnected from the Internet.

Because EC offers relatively high computational power close to the end-users, it is well suited for offering real-time, critical, local, private services to citizens [4].

A.5 Mobile Edge Computing

MEC is a variant of EC. The main difference between the two paradigms is that MEC focusses on mobile devices and the Radio Access Network (RAN). Computing nodes are located at the Base Stations (BSs) of the RAN edge to provide computing capabilities and services close to the end-users [5].

B. E2CaaS architecture for Smart Cities

To form a baseline for the proposed architecture for the ZEN Research Center in III-A, the author of this study has proposed an Edge-to-Cloud-as-a-Service (E2CaaS) architecture for Software Services in Smart Cities in [6]. This architecture places the concepts explained in the previous section inside the scenario of a Smart City. The idea behind this architecture is based on a literature review discussed in the next section and the previous work of *Sinaeepourfard et al.* [7].

C. Developing Software Services in Smart Cities

Next, as the results of a literature review on FC, EC, and cloudlets in Smart Cities, and a systematic literature review by *Javadzadeh et al.* [8], the author has proposed a model on how to develop Software Services in Smart Cities. The model, shown in Fig. 1 consists of four main steps: Classification of City Services, Design Output, Implementation, and Efficiency Measurements.

The goal of the Classification of City Services is to define the Domain of the service inside the city. Alongside this domain, some (non-)functional requirements can be identified. It is also important to consider the City and ICT Objectives in this step.

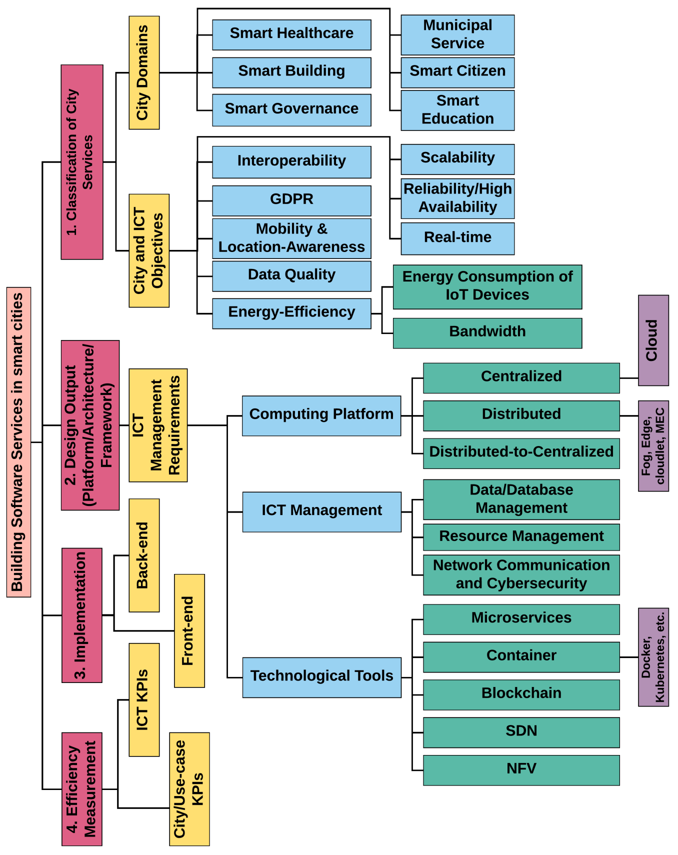


Fig. 1: Developing Software Services in Smart Cities

The figure shows some common non-functional requirements for a Software Service in a Smart City.

Next, in the Design Output, the type of Computing Platform, the ICT Management strategies, and the Technological Tools are selected. Some possible Computing Platforms, such as a Centralized and Distributed Computing Platform, are shown in the illustration. The three main categories of the ICT Management that should be considered are the Data/Database Management, the Resource Management, and the Network Communication Management and Cybersecurity. The figure also shows some common Technological Tools that were encountered during the literature review and that can execute the different tasks that come along with the ICT Management.

In the third step, the Implementation, the selected technologies and tools are implemented. This often results in a Front-End and a Back-End. The Back-End contains most of the application logic and algorithms. The Front-End is used by the clients, citizens, or employees, to use the service. Both parts of the application can interact with each other through an Application Programming Interface (API) or Software Development Kit (SDK).

In the final step, the Efficiency Measurements are taken and analyzed. These measurements come in the shape of Key Performance Indicators (KPIs). A KPI is a measurable value that indicates how well the service achieves key business objectives. These values can be obtained by performing simulations, measurements, and surveys on the service [9]. Two types of KPIs

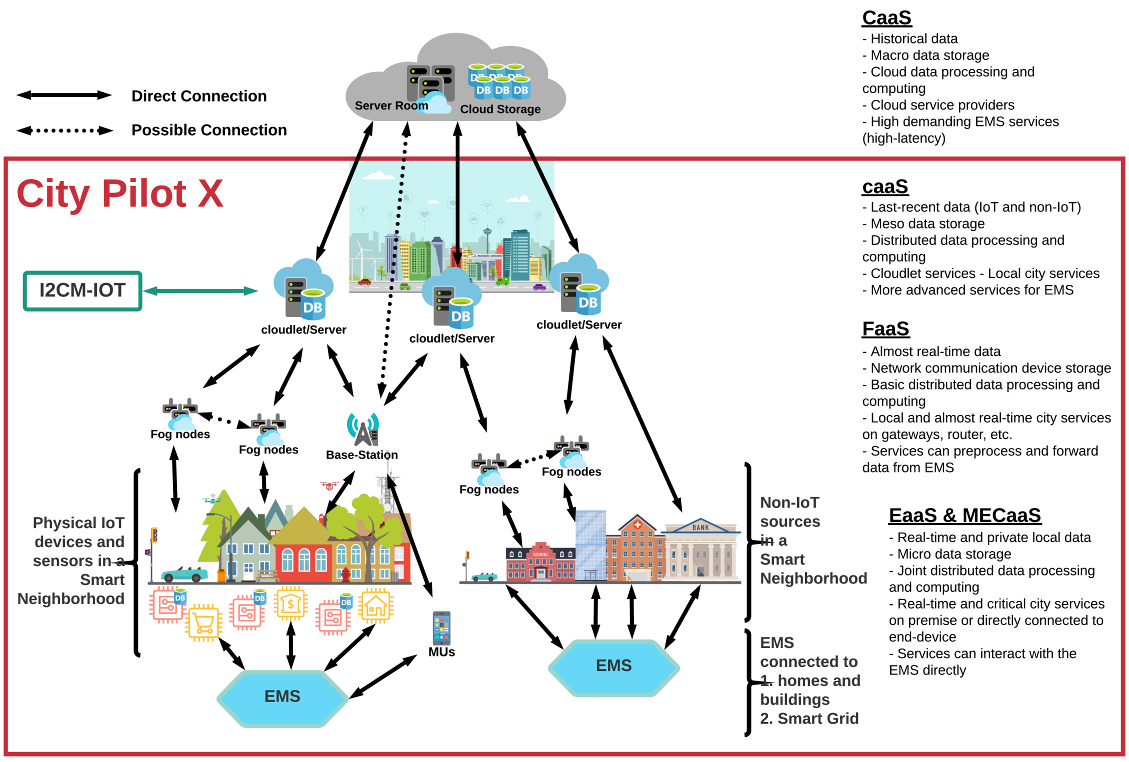

Direct Connection
Possible Connection
Server Room
Cloud Storage
DB DB DB
CaaS
- Historical data
- Macro data storage
- Cloud data processing and computing
- Cloud service providers
- High demanding EMS services (high-latency)
City Pilot X
I2CM-IOT
cloudlet/Server
cloudlet/Server
cloudlet/Server
DB
caaS
- Last-recent data (IoT and non-IoT)
- Meso data storage
- Distributed data processing and computing
- Cloudlet services - Local city services
- More advanced services for EMS
FaaS
- Almost real-time data
- Network communication device storage
- Basic distributed data processing and computing
- Local and almost real-time city services on gateways, router, etc.
- Services can preprocess and forward data from EMS
Fog nodes
Fog nodes
Base-Station
Fog nodes
Fog nodes
Physical IoT devices and sensors in a Smart Neighborhood
Non-IoT sources in a Smart Neighborhood
MUs
EMS
EMS
EMS connected to 1. homes and buildings 2. Smart Grid
EaaS & MECaaS
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical city services on premise or directly connected to end-device
- Services can interact with the EMS directly

Fig. 2: E2CaaS architecture in a ZEN Center pilot

are considered in this study: ICT KPIs and city/use-case KPIs. The former are general measurable values like bandwidth and latency [7]. These KPIs are related to the Data/Database Management, the Resource Management, and the Network Communication Management and Cybersecurity. The latter depend more on the requirements of the Smart City or the specific domain/use-case of the service.

More details on this model for Smart Cities are published in [6]. The model is used in Section III to discuss developing Software Services for EMSs, as a use case for the ZEN Research Center.

III. EMS Software Services in Smart Neighborhoods

The ZEN Research Center is a research center located in Norway that researches Zero Emission Neighborhoods (ZEN) in Smart Cities. The main goal is to contribute to a low carbon society by achieving no emission of greenhouse gas in the neighborhoods of Smart Cities[1]. This research is conducted through eight pilots in Norway, located in Bodø, Trondheim, Steinkjer, Evenstad, Elverum, Oslo, Bærum, and Bergen [10].

One of the interests of the ZEN Center is to develop an architecture for building efficient Software Services for Smart Cities and Smart Neighborhoods. This architecture can be used to develop EMS services that can reduce the energy consumption and improve the Quality of Life in the city and its neighborhoods.

This section aims to address this goal by applying the knowledge, gained from designing the architecture and model in the previous section, onto the ZEN Research Center as a use-case.

A. E2CaaS architecture for the ZEN Center

The result of applying the proposed E2CaaS architecture to the pilots of the ZEN Center is shown in Fig. 2.

The IoT devices are located inside the Smart Buildings and Smart Homes. These devices, which can be sensors, smart meters, actuators, etc., are connected to the EMSs and generate data from the Smart Grid. This data can be used by Software Services running somewhere in the neighborhood, city, or Cloud. For private, critical, or real-time applications, the citizens can appeal to the Edge nodes inside or near-by the Smart Homes and Smart Buildings. These devices offer relatively high computing power and can work independently. For monitoring and managing their EMS, the citizens can use their mobile devices, such as smartphones, that are connected to the Radio Access Network (RAN) and can offload computing tasks using MEC. The data can also be sent towards the Fog nodes. These nodes are gateways and routers with limited computational capabilities that can preprocess and forward the data to the stronger cloudlets and Cloud. The Cloud can be used for applications running across multiple pilots or for high demanding services that need a lot of computing power and storage.

The non-IoT datasources, shown on the bottom right of the figure, produce data based on human-human and human-machine interactions. The companies and organization which produce this kind of data can have their own cloudlets at their disposal. Consequently, these datasources are directly connected to the cloudlets. If this is not the case, Fog nodes can be used to process and/or forward the data to the cloudlets and

[1]https://fmezen.no/

Cloud.

After some processing task is executed on the data, the services can report back to the citizens and to the EMSs to give feedback and to improve the energy management in the city.

As with the general E2CaaS architecture, the goal is to move the control over the computing and network resources to the cloudlet layer. This idea, introduced by *Sinaeepourfard et al.* [7] uses an Integrated and Intelligent Control and Monitoring of IoT (I2CM-IoT) box which is responsible for the ICT management inside the city.

B. Developing EMS Software Services in Smart Neighborhoods

This section adapts the model on how to develop Software Services in Fig. 1 to the ZEN Research Center. The adapted model can be seen in Fig. 3. Again, the same four main categories are identified in this model.

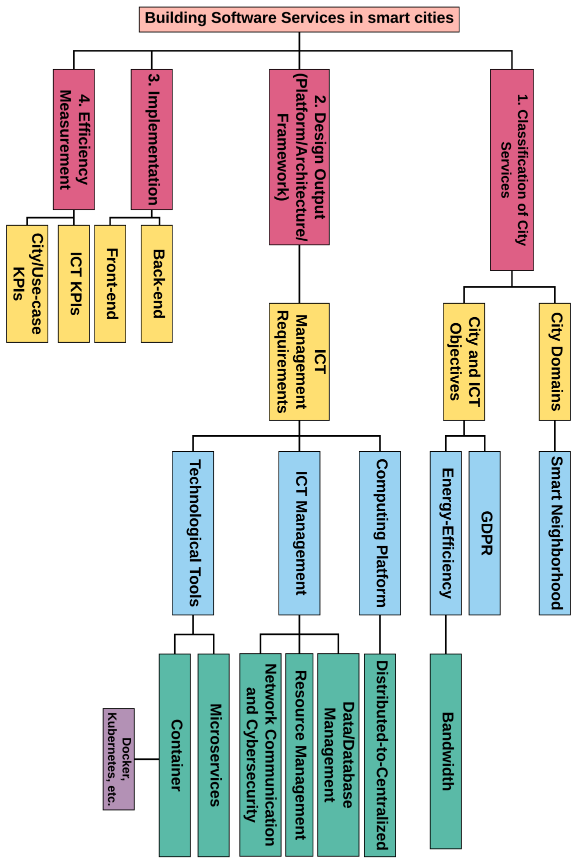


Fig. 3: Developing EMS Software Services for Smart Neighborhoods

B.1 Classification of City Services

For the Classification of City Services, the focus lies on the Smart Neighborhood and EMS Domain. This means that the services are located close to the citizens, and will process private data from the citizens. Additionally, the services will work together with an EMS, thus should be energy efficient. Because of these requirements, two main City and ICT Objectives are identified: the General Data Protection Regulation (GDPR) and Energy Efficiency in terms of the bandwidth usage. Notably, more objectives can be applied to this scenario. The two selected objectives serve as an example to show the reader how the model should be used.

B.2 Design Output

When designing the output, like in the general architecture, the Computing Platform, the different ICT Management strategies, and the Technological tools should be selected. The first two categories of the Design Output are discussed in this section. The chosen Technological Tools are explained in Chapter IV.

In terms of Computing Platform, the ZEN Center is mainly interested in a D2C platform. This type of platform enables both privacy-friendly and real-time computation at the edge of the Smart City and computing-intensive applications in the Cloud.

Next, the different ICT Management categories are discussed briefly.

- In terms of the database selection for the **Database Management**, there are many options to choose from. Because the database type that should be used depends on the specific services that will be implemented, this study does not suggest an "optimal" database.

Sinaeepourfard et al. [9] defined a data management architecture for the ZEN Center scenario. The authors have identified three types of data: Context Data, Research Data, and KPI Data. The context data is coming from the datasources inside the Smart Neighborhoods. This data is not useful on its own but can be processed to gain valuable knowledge and information. Context Data is available on all levels of the E2CaaS architecture. Research Data is generated by special dedicated applications such as simulations or data planning applications. Research Data is available on the Meso (cloudlet) and Macro (Cloud) level. This data is produced by the Smart Buildings in the pilots. The final type of data, KPI Data, is gathered based on the predefined KPIs of the ZEN Center. This data is collected by carrying out surveys, simulations, and measurements. The KPI Data can be found, again, on all levels.

- The **Resource Management** for the use-case is similar to that of the general model for Smart Cities. The orchestration is placed inside the cloudlet layer using the proposed I2CM-IoT box [7]. This could be realized using a container orchestrator located on the cloudlets. This orchestrator can then proceed to manage the resources inside the neighborhoods of the Smart City. The disadvantage of container orchestrators like Kubernetes is that they need their control plane to be reliable and highly available. Resulting in most systems placing their orchestrator in a centralized, Cloud-based location. A possible solution is to divide the city into smaller clusters, each managed by a cloudlet. Another approach could utilize Software-Defined Networking (SDN) controllers for managing the resources. However, as with container orchestrators, the programmable control plane of SDN is centralized.
- As with the selection of the database, there are many available technologies for the **Network Communication Management and Cybersecurity**. Each of these technologies have their advantages and disadvantages and thus should be selected based on the service. For large-scale, real-time, and critical services,

messaging queue technologies such as Apache Kafka Streams[2] or the various MQTT Brokers are a good option. Simple applications that do not require asynchronous communication and high-scalability can use other methods such as HTTP Requests. Security and privacy between the computing nodes can be accomplished by technologies such as Blockchain and SDN. For securing the EMS of Smart Homes and Smart Neighborhoods, other proposals have been made. For example, a study by *Mugarza et al.* [11] proposes a solution for security in EMS Smart City applications using the Cetratus framework [12] for enabling Dynamic Software Updates (DSUs).

B.3 Implementation

The frameworks and tools for developing the Front-End and Back-End depend on the preferences of the developers and of the actual purpose of the service.

A Front-End for managing ZEN KPIs is already introduced by *Sinaeepourfard et al.* in [7].

B.4 Efficiency Measurements

The ZEN Center has defined seven categories with sets of assessment criteria and KPIs for achieving zero emission neighborhoods. These seven categories are Greenhouse Gas Emission, Energy, Power/Load, Mobility, Economy, Spatial Qualities, and Innovation. *Sinaeepourfard et al.* [7] place these criteria and KPI s inside a "ZEN KPI box." This box is then connected to the "ZEN Toolbox," presented in [13]. These two boxes work together with the I2CM-IoT control entity in the cloudlet layer to measure the performance and assess the Software Services of a specific pilot.

IV. Implementation of a Software Service

The goal of this final chapter is to give a simplified example on how to develop a Software Service for an EMS in a Smart Neighborhood or Smart City. The implementation will focus on containerization for managing the resources and the deployment of the application. Four different scenarios are implemented to visualize the impact of moving the computing tasks (partially) closer to the edge of the network.

A. Scenario and Setup

The service that will be implemented uses the proposed E2CaaS architecture. The service reads temperature values from sensors of an EMS in a Smart Home or Smart Building. A simple processing job is executed on the temperature values. In a real-world scenario, this could provide valuable information to the clients. The service can be extended easily by, for example, sending feedback and notifications to the citizens and EMS when the temperature values reach a specific threshold.

For the **Data/Database Management**, the best choice for this kind of scenario would be a lightweight MySQL database, as the temperature values are simple consistent values. For more complex or variable data, NoSQL databases that use a document-like structure can be used.

As mentioned before, containerization is used for the **Resource Management**. The container technology product of choice for this implementation is Docker[3]. Docker is one of the most popular containerization platforms that offers easily deployable and manageable containers. Plus, it works well with the chosen container orchestrator: Kubernetes[4]. Kubernetes is also widely used and enables developers to automate the deployment, scaling, and management of containerized applications, making it an excellent choice for managing the resources. However, it is not possible with a default Kubernetes setup to move the control plane to the distributed cloudlet layer, as is suggested in the E2CaaS architectures. In future work, alternatives such as KubeEdge should be considered to solve this problem.

In terms of the **Network Communicationn Management**, Kubernetes is compatible with a lot of plugins that enable network communication between cluster nodes. This study uses the Weave Net plugin[5]. This plugin is easy and quick to deploy and does not require any configuration. However, the Weave Net plugin does not come with any privacy and security features out of the box. Consequently, the **Cybersecurity** should also be addressed in the future.

The data is sent from node to node using HTTP POST requests. This method is widely used and easy to set up. The author suggests that future work looks into publish/subscribe protocols such as Message Queuing Telemetry Transport (MQTT) instead. These types of protocols are more suited for sending real-time data across the network, as they offer asynchronous communication and high scalability.

B. Results

The application is deployed on the imec IDLab Virtual Wall. Four different scenarios are implemented: the first scenario uses 1 IoT node, 1 cloudlet, and 1 Cloud node; the second scenario uses 2 IoT nodes, 1 cloudlet, and 1 Cloud node; the third scenario uses 4 IoT nodes, no cloudlets, and 1 Cloud noded; the final scenario uses 4 IoT nodes, 2 cloudlets, and 1 Cloud node.

Fig. 4a shows the average CPU usage of the cloudlet nodes when increasing the maximum requests per second on the IoT devices. Adding an extra IoT devices to a cloudlet increases the load on the cloudlet by 93%. This is to be expected as the device has to process twice as many requests. Notably, the CPU values stagnate from around 2 to 3000 requests/s. This is caused due to the IoT devices not being able to send more requests/s.

Fig. 4b shows that moving the computing tasks closer towards the citizens can reduce the load on the Cloud significantly. However, it is notable that when using four IoT nodes and two cloudlets, the test did become more variable. The readings from the cloudlets are not as constant compared to the other scenarios.

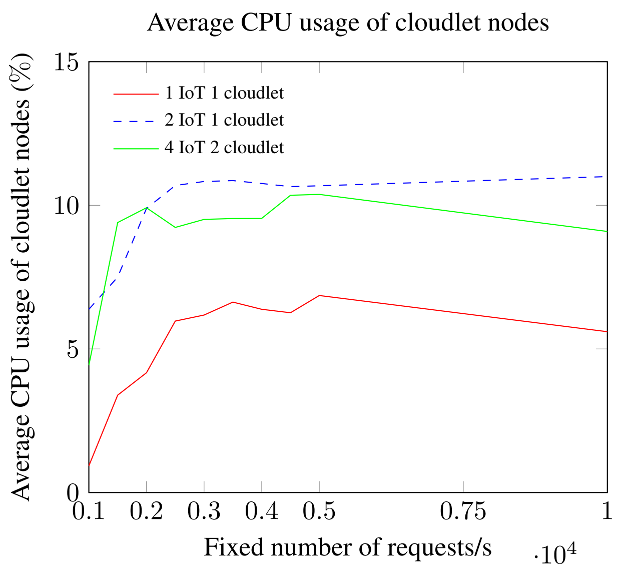
By measuring the maximum number of requests per seconds at which the IoT devices were able to send their requests in the different scenarios shows that using two and four IoT devices reduces the individual performance of the devices respectively by 8% and 19%. But when taking into account that there are now multiple devices sending data simultaneously, an overall increase in performance of 84% and 225% is observed.

[2]`https://kafka.apache.org/documentation/streams/`

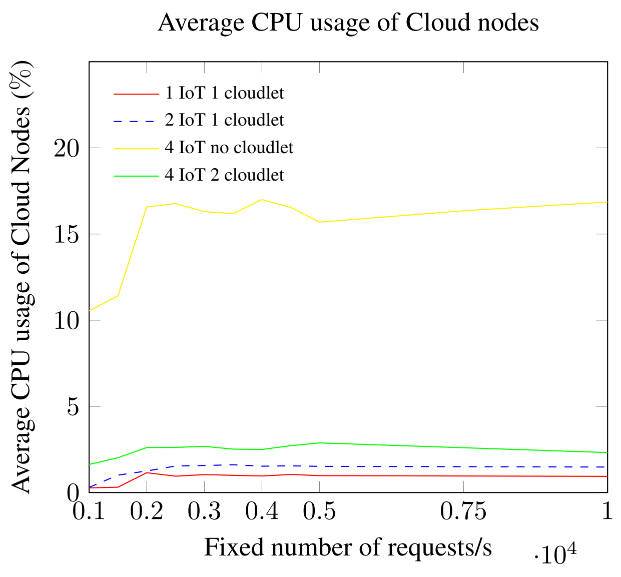
[3]`https://www.docker.com/`

[4]`https://kubernetes.io/`

[5]`https://www.weave.works/oss/net/`



(a) Average CPU usage of cloudlet nodes



(b) Average CPU usage of Cloud nodes

Fig. 4: The average CPU usage of cloudlet and Cloud nodes in function of the increasing fixed number of request/s

V. Conclusion

This study proposed an E2CaaS architecture and model for developing Software Services in Smart Cities based on a literature review. Afterward, this architecture and model are applied onto the ZEN Research Center to provide a base-line for developing EMS Software Services for Smart Neighborhoods. A simplified Software Service is implemented and deployed on the imec IDLab Virtual Wall. Measurements show that it is indeed beneficial to move the computing task closer to the citizens, on a distributed platform. This study does not aspire to give a complete overview of all the available technologies and possibilities regarding Software Services. Alternatively, the author seeks to lay a foundation and give a general direction for the future of Software Services in Smart Cities and the ZEN Center.

Future work should focus on technologies that enable the control over the network, data, and resources on the cloudlet layer. Alternatives to Kubernetes, such as KubeEdge, could be worth looking into. Additionally, using a publish/subscribe messaging queue protocol such as MQTT could significantly improve the performance and scalability of the service. Finally, the implemented scenario did not pay any attention to security and privacy issues regarding the EMS Software Services.

Acknowledgement

This study has been carried out within the Research Centre on Zero Emission Neighborhoods in Smart Cities (FME ZEN). The author gratefully acknowledges the support from the ZEN partners and the Research Council of Norway. Additionally, the author wants to thank his supervisors, co-supervisor, and councilor for their continuous support and advice.

References

[1] A. G. Prajapati, S. J. Sharma, and V. S. Badgujar, "All About Cloud: A Systematic Survey," in *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, Jan. 2018, pp. 1–6.

[2] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[3] R. Huang, Y. Sun, C. Huang, G. Zhao, and Y. Ma, "A Survey on Fog Computing," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, ser. Lecture Notes in Computer Science, G. Wang, J. Feng, M. Z. A. Bhuiyan, and R. Lu, Eds. Cham: Springer International Publishing, 2019, pp. 160–169.

[4] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, Sep. 2019.

[5] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[6] J. Robberechts, A. Sinaeepourfard, T. Goethals, and B. Volckaert, "A Novel Edge-to-Cloud-as-a-Service (E2CaaS) Model for Building Software Services in Smart Cities," in *IEEE International Conference on Mobile Data Management (MDM 2020)*, Versailles, France, Jun. 2020.

[7] A. Sinaeepourfard, J. Krogstie, and S. A. Petersen, "A Distributed-to-Centralized Smart Technology Management (D2C-STM) model for Smart Cities: A Use Case in the Zero Emission Neighborhoods," in *2019 IEEE International Smart Cities Conference (ISC2)*, Oct. 2019, pp. 760–765.

[8] G. Javadzadeh and A. M. Rahmani, "Fog Computing Applications in Smart Cities: A Systematic Survey," *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, Feb. 2020.

[9] A. Sinaeepourfard, J. Krogstie, S. A. Petersen, and D. Ahlers, "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Apr. 2019, pp. 590–595.

[10] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Data Preservation through Fog-to-Cloud (F2C) Data Management in Smart Cities," in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, May 2018, pp. 1–9.

[11] I. Mugarza, A. Amurrio, E. Azketa, and E. Jacob, "Dynamic Software Updates to Enhance Security and Privacy in High Availability Energy Management Applications in Smart Cities," *IEEE Access*, vol. 7, pp. 42 269–42 279, 2019.

[12] I. Mugarza, J. Parra, and E. Jacob, "Cetratus: Towards a live patching supported runtime for mixed-criticality safe and secure systems," in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, Jun. 2018, pp. 1–8.

[13] A. H. Wiberg and D. Baer, "ZEN TOOLBOX: First concept for the ZEN Toolbox for use in the development of Zero Emission Neighbourhoods Version 1.0," NTNU/SINTEF, Memo, 2019.

1 Introduction

1.1 Background

With Smart Cities becoming more potent in recent years, the need for an efficient approach to develop Software Services for improving the Quality of Live of the citizens and the sustainability of the city is urging. Alongside this evolution, novel technologies related to Internet of Things (IoT) and data-centric applications inside cities are imminent. This thesis aims to explore and bring some structure to this vast ocean of technologies and methodologies that are available from the Edge of the network to the Cloud, mainly focusing on the different multilevel technologies such as *Edge Computing (EC), Fog Computing (FC), cloudlets*, and *Cloud Computing (CC)*. Some Research Questions are identified to provide a clear objective throughout this work. With the help of a broad literature review, the reader is guided through the process of developing Software Services in Smart Cities. Proceeding towards this goal, many technologies are explained and discussed. Additionally, the knowledge gained from this literature study is applied to the ZEN Research Center. The ZEN Center researches zero emission of greenhouse gas in Smart Neighborhoods. Software Services can help to achieve this objective by managing Energy Management Systems (EMSs) in these neighborhoods. Notably, this study does not aim to give a complete overview of all the possibilities regarding Software Services in Smart Cities. Alternatively, the author seeks to lay a foundation and give a general direction for the future of Software Services in Smart Cities and the ZEN Center.

1.2 Research Questions

The following research questions have been identified for this thesis:

- Q1: How to develop efficient Software Services using multilevel ICT architectures from Edge to Cloud in Smart Cities?
  - Q1.1: What are the current technological trends for developing Software Services in Smart Cities from Edge to Cloud?
  - Q1.2: How to use these technologies to build Software Services from Edge to Cloud?
- Q2: How to build EMS Software Services from Edge to Cloud in Smart Neighborhoods?
  - Q2.1: What is a possible architecture for developing EMS Software Services in Smart Neighborhoods that uses the discovered technologies and methods from Q1.2?
  - Q2.2: How to develop an EMS Software Service using this proposed architecture while focusing on Container Technologies?

1.3 Approach and Outline

This section goes over the approach and methodology that was used when addressing the research questions defined in the previous section.

In the first part of this thesis (Chapter 2), some background knowledge about the concepts *Cloud -, Edge* - and *Fog Computing* is gathered using general literature reviews and state-of-the-art surveys on the subjects. These concepts are explained in Section 2.1. Afterward, a literature review on *Edge* - and *Fog Computing* in Smart Cities is conducted to solve *Q1.1.* This literature review is discussed in Section 2.2. With the knowledge gathered from this literature study, *Q1.2* is solved by proposing a method for developing Software Services in Smart Cities.

The second part of this thesis can be found in Chapter 3 and aims to solve *Q2.1* by applying the knowledge from the general study onto the ZEN Research Center as a use-case. This results in an architecture for developing EMS Software Services for the ZEN Center. Afterward, to validate the viability of the proposed architecture and to answer *Q2.2*, a simplified example of an EMS Software Service is developed using the proposed architecture. The process of developing this example, the tools that were used, and the results are discussed in Chapter 4.

Finally, some reflections, thoughts, and future work are given in Chapter 5.

All the work and tasks that are stated in this section were carried out as a collaboration between the ZEN research center, the Norwegian University of Science and Technology, and Ghent University.

1.4 Publications

In the process of writing this Master's Thesis, a compact version of Chapter 2 is published at the **The First International Workshop on (3SCity-E2C) Building Software Services in Smart City through Edge-to-Cloud orchestration**[1] in conjunction with the **21st IEEE International Conference on Mobile Data Management**[2], Versailles, France (Online Conference). This paper can be found in Appendix A.

J. Robberechts, A. Sinaeepourfard, T. Goethals and B. Volckaert, ***"A Novel Edge-to-Cloud-as-a-Service (E2CaaS) Model for Building Software Services in Smart Cities,"*** *2020 21st IEEE International Conference on Mobile Data Management (MDM), Versailles, France, 2020, pp. 365-370*

[1]`https://fmezen.no/3scity-e2c-workshop-2020/`
[2]`http://mdmconferences.org/mdm2020/`

2 Software Services from Edge to Cloud in Smart Cities

This chapter starts by explaining the different service models and technologies that can be used to offer services to citizens in Smart Cities. Next, it covers a literature review on some of these models. Afterward, a model on how to develop software services from edge to cloud in Smart Cities is proposed using the knowledge extracted from this literature review.

2.1 Background

This section explains some vital concepts regarding Software Services in Smart Cities. These concepts include *Smart Cities, Cloud -, Fog -, Edge -, and Mobile Edge Computing*.

2.1.1 Smart Cities

Recently, many devices and concepts are acquiring the term “smart”. Smart devices, for example, are devices that can communicate with each other and perform some kind of task more or less independently, without human interaction. Another type of device that gets more attention lately, is IoT devices. These IoT devices can be smart devices that contain a sensor or actuator to measure or observe something. Together, IoT and smart devices can transform a city into a Smart City. They enable new services that can enhance the Quality of Life inside the city or improve the sustainability of the city [1, 2]. Software Services play an important role in Smart Cities. There is a constant demand for new and better services to further improve the Quality of Life in the city. One example of such a smart service is a smart car. Lately, smart vehicles are getting more numerous because they offer features and tools to the citizens which improves their driving experience. The government, organizations, and companies are also interested in Software Services. Based on data gathered from the citizens and IoT devices they can improve their existing services or introduce new services for the customers and citizens.

2.1.2 Cloud Computing

According to [3], *Cloud Computing (CC)* is a computing model where users can utilize computing resources on-demand, managed by a Service Provider (SP). This means that the client does not have to invest in expensive dedicated hardware for computing tasks and software services. Instead, the client can rent the necessary resources from the SP. There are three common methods for providing *Cloud-as-a-Service (CaaS)*: infrastructure providers who provide *Infrastructure-as-a-Service (IaaS)*, platform providers providing a *Platform-as-a-Service (PaaS)*, software providers who provide *Software-as-a-Service (SaaS)*, and providers who charge their clients based on their usage, which is called *Serverless Computing*. These different concepts will be explained in detail in the next paragraphs and can be seen in Fig. 2.1.

IaaS offers the client virtualization, storage, and processing [5]. The clients have a lot of freedom, they can, for example, choose the Operating System (OS) and software that they want to use for their tasks. Fig. 2.1 shows that the model

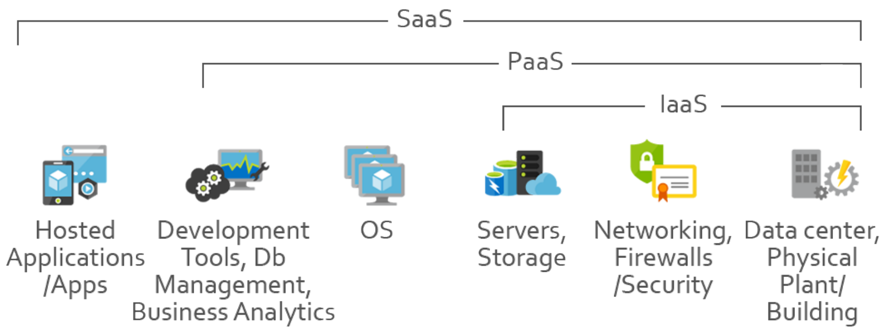

SaaS
PaaS
IaaS
Hosted Applications /Apps
Development Tools, Db Management, Business Analytics
OS
Servers, Storage
Networking, Firewalls /Security
Data center, Physical Plant/ Building

Figure 2.1: IaaS, PaaS, and SaaS [4]

includes the management of the datacenter, the management of the networking and security between the nodes in the cloud, and the management of the cloud servers and storage. *IaaS* is primarily used for product design, data storage, website hosting, big data analysis, and temporary processing campaigns [4, 5]. Examples of companies that offer *IaaS* are Amazon AWS and Microsoft Azure IaaS.

PaaS clients do only have control over their applications. They do not have to manage the underlying infrastructure, OS, and development tools [5]. Fig. 2.1 shows that the *PaaS* method includes the aspects of *IaaS*, and also offers the OS and the development tools, database management, and business analytics. Compared to *IaaS*, the client has less freedom but can still decide on the application itself. This makes the development of applications on *PaaS* easier and quicker. This service model is mostly used as a development framework, or for analytics and additional services [6]. Some common examples are Microsoft Azure, Salesforce, and Amazon AWS.

SaaS providers offer applications that are running on the cloud. Consequently, the client has no control over the infrastructure, platform, and application whatsoever. The clients can connect to the applications by connecting to the internet through, for example, a browser. The main advantage of *SaaS* is that it allows the client to be quickly up and running with the least amount of effort and minimal upfront cost [7]. Well known examples of *SaaS* are Facebook, Google, and Twitter.

Serverless Computing offers less flexibility than PaaS as the client does not have to create the whole application. The client does get more freedom than SaaS as single functions of the application can still be created.

2.1.2.1 Cloud Computing in Smart Cities

As the goal of a Smart City is to enhance services and to increase the quality of life inside the city through smart technologies, *CC* is used as a tool to manage, store, and process the data coming from the different ICT devices inside the city [8]. The *Cloud* can be used by the government and by organizations to improve the quality of their services, which are in turn used by the citizens to improve their Quality of Life. Some of the benefits of using *CC* in Smart Cities are reduced cost of services for the citizens, more agile, flexible, and elastic services, globally available services, and support for sustainable development [5].

2.1.3 cloudlet Computing

cloudlets, introduced by *Satyanarayanan et al.* in 2009 [9], are a concept that places the computing power closer towards the end-users and datasources. A *cloudlet* can be a small datacenter or group of computing devices. Many studies refer to the *cloudlet* as a datacenter in a box whose goal it is to move the *Cloud* closer towards the edge of the network.

The main advantages of moving the computing power to these smaller datacenters is improved latency for the citizens, and possibly improve the security and privacy. The downside to *cloudlets* are additional complexities due to handovers, which occurs when a mobile user is moving and has to be transferred from one *cloudlet* to another, and the need for rapid provisioning. Table 2.1 shows the difference with some features of *cloudlet* and *Cloud*.

2.1.4 Fog Computing

Fog Computing (FC) is a concept that was designed to counter the weaknesses of *CC*. There are many issues concerning *CC* especially when handling critical IoT services. Some of these issues are high latency and lack of location awareness due to the *Cloud* being located far away from the datasources, and higher risk of security breaches while the data is being transferred towards the *Cloud*. The *Fog*

Table 2.1: Distinct features of cloudlet - and Cloud Computing

Features	**cloudlet**	**Cloud Computing**
Service node location:	Local datacenter in city	Cloud datacenter
Number of nodes:	Medium	Low
Delay:	Medium	High
Bandwidth requirement:	Medium	High
Computing capability:	Medium-High	High
Mobility support:	Some mobility support	No
Service type:	Local	Global
Architecture:	Decentralized	Centralized

paradigm solves these issues by moving the computing towards the edge of the network. Thereby enabling real-time or near real-time data processing, possibly offering better data privacy (e.g. federated learning), and offering computing on widely distributed geographical locations [10].

If we describe the *Cloud* as a centralized unit of computing, the *Fog* can be seen as a network of small devices with limited computing capabilities that work together in a decentralized way. Some characteristics of *Fog* and *Cloud* are compared in Table 2.2. *Fog Nodes* are often grouped inside different domains, each managed by an SP. The nodes are working together but are controlled and provisioned by an entity (e.g. master node or fog leader) belonging to the SP of their domain [11, 12]. The different *Fog Domains* can work together and offload tasks to each other, thus forming a distributed network of computing nodes. *Fog Nodes* include many different types of devices like routers, smartphones, and wireless access points.

Because *Fog Nodes* do not have a lot of computing power at their disposal, they are often used for small computing tasks such as data preprocessing. This reduces the data that has to be sent towards the datacenters in the *Cloud* or *cloudlet*.

2.1.5 Edge Computing

The terms *Fog* and *Edge Computing (EC)* are often used as synonyms. Although these concepts are similar, they are not the same. Both paradigms try to extend the *Cloud* towards the edge of the network. But where *FC* moves the computing of the *Cloud* downwards, *EC* uses the computing power available at the edge of the network to offer services. The computing takes place either on the IoT device,

Table 2.2: Distinct features of Fog - and Cloud Computing [10]

Features	**Fog Computing**	**Cloud Computing**
Service node location:	Between source and center	Cloud datacenter
Number of nodes:	Very High	Low
Delay:	Low	High
Bandwidth requirement:	Low	High
Computing capability:	Low	High
Mobility support:	Yes	No
Service type:	Local	Global
Architecture:	Distributed/Decentralized	Centralized

where the data is generated, or on some computing device or local datacenter close to the source [13]. *FC* uses gateways and other devices with limited computing power, making it highly dependent on more centralized cloud-like technologies, whereas *EC* can work more independently and even disconnected from the Internet.

Because the computing power is located as close as possible to the data sources and citizens, *EC* can offer critical real-time services, privacy and high connectivity [13]. As with *FC*, this technique is also able to offer services in widely distributed geographical locations. Some characteristics of *Fog* and *Cloud* are compared in Table 2.2.

Table 2.3: Distinct features of Edge - and Cloud Computing

Features	**Edge Computing**	**Cloud Computing**
Service node location:	Between source and center	Cloud datacenter
Number of nodes:	Very High	Low
Delay:	Very Low (Real-time)	High
Bandwidth requirement:	Low	High
Computing capability:	Low-Medium	High
Mobility support:	Yes	No
Service type:	Local	Global
Architecture:	Decentralized	Centralized

2.1.6 Mobile Edge Computing

Mobile Edge Computing (MEC) is a similar concept to *Edge Computing*, except that it focuses on the Radio Access Network (RAN). *MEC* uses computing devices at the Base Stations (BSs) of the RAN edge to provide computing capabilities and services close to the end-users [14, 15].

The main advantages of *MEC* to the user are the ability to offer location-based applications, and the potential for a variety of new applications using contextual data and content. The SPs, often Mobile Network Operators (MNOs), benefit from *MEC* by collecting more information on their customers in terms of location, interests, and content which can be used to improve their quality of service or to offer new types of applications [14].

MEC is mostly used for data pre-processing, task-offloading from mobile devices, and context-aware services. Some features of *MEC* are compared to *CC* in Table 2.4 [15].

Table 2.4: Distinct features of MEC - and Cloud Computing

Features	**MEC**	**Cloud Computing**
Service node location:	Between source and center	Cloud datacenter
Number of nodes:	Very High	Low
Delay:	Very Low (Real-time)	High
Bandwidth requirement:	Low	High
Computing capability:	Low-Medium	High
Mobility support:	Yes	No
Service type:	Local	Global
Architecture:	Distributed	Centralized

2.2 Literature Review

To get a better view on current technological trends involving *FC* and *EC*, a general literature review was carried out. This study aims to solve research question *Q1.1* and to lay a foundation for research question *Q1.2*.

2.2.1 Methodology

The search engines used for the literature review are *IEEE Xplore*[1] and *Scopus*[2]. These engines were selected because they offer a wide variety of options that are easily adjustable based on the needs of the researcher. They both handle citations well and contain a rich collection of published work.

The query that is used for this research can be found in Fig. 2.2. As the scope of this query is quite large and the literature had to be studied in a short period, the results were narrowed down to the last quarter of 2019 and the first quarter of 2020. This resulted in 247 Journal Papers and 46 Conference Papers.

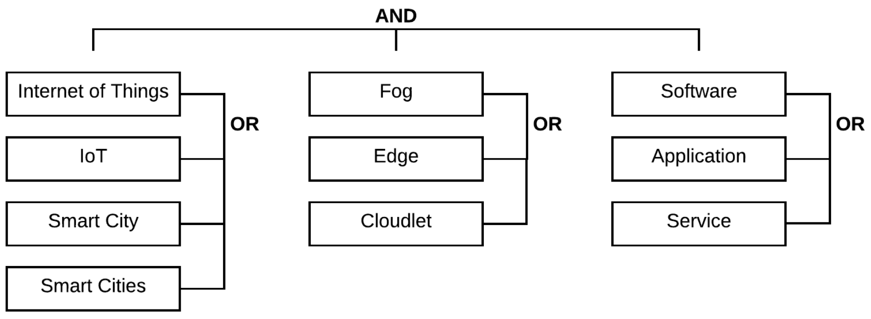


Figure 2.2: Literature Review Query

A first selection of 76 papers was made based on the title and abstract from the list of 293 papers. This selection was narrowed down to 20 papers based on their quality and content. The next subsection will go over what there is to learn from this selection of papers.

2.2.2 Result

Most of the selected papers go into detail on one specific aspect of Software Services in Smart Cities by either addressing a problem related to the ICT management or by addressing or trying to improve one scenario of a domain in a

[1]`https://ieeexplore.ieee.org/`
[2]`https://www.scopus.com/`

Smart City. These studies often do not discuss in detail how their services are being deployed.

To to solve research questions *Q1.1* and *Q1.2*, and to resolve the issue highlighted above, a model on how to develop Software Services in Smart Cities is proposed in Section 2.4 (Appendix A). This model can be seen as a summary of the work that is studied in this literature review, as it features the most common technological trends that were encountered. The goal of this model is to give a general direction when developing Software Services using different multi-level technologies in Smart Cities. The selected papers from the literature review are classified based on this model to provide some examples of proposals and technologies to the reader.

2.3 Edge-to-Cloud-as-a-Service architecture for Smart Cities

Because most of the current studies do not focus on the use of a combination of different multilevel distributed and centralized technologies and to form a baseline for *Q2.1*, an *Edge-to-Cloud-as-a-Service (E2CaaS)* architecture for Smart Cities is proposed by the author of this thesis (Appendix A). The idea behind this model is to use cloudlets as a middleware layer between the *Cloud* and the edge of the City. An overview of this city architecture is shown in Fig. 2.3. This image shows how all these different distributed and centralized technologies have their own place inside a Smart City.

The top layer is the *Cloud-as-a-Service (CaaS) layer*. The *Cloud* is the largest entity in the architecture and covers more than the Smart City on its own. This layer houses the historical data, as discussed in Section 2.4.4.1 about data management. This data is not produced right now but is being stored for later use or services that do not need fast response times. The *Cloud* has almost unlimited resources, resulting in Macro data storage and high demanding computation tasks. The second layer, *cloudlet-as-a-Service (caaS)*, is located inside the city, closer to the citizens. The *cloudlets* contain the last-recent data and still have decent storage and computing capabilities. They also offer services with lower latency. Plus, the data does not have to leave the city, which improves the security and privacy of the citizens. *Fog-as-a-Service (FaaS)* offers fast response times close to the citizens in a distributed fashion. These kinds of devices are often used for simple preprocessing tasks. Next, the bottom right of the figure, contains the non-IoT data sources. Non-IoT data is data resulting from human-device or human-human interactions. This kind of data can be produced by companies or organizations. The IoT data sources and *Edge Nodes* are located in the middle of the lowest layer. *Edge-as-a-Service (EaaS)* can offer private, local, real-time, critical services close to the citizens with decent computing capabilities. These devices can be dedicated computing nodes in shops or near houses of citizens. Finally, the left side shows *MECaaS*. This model offers mobile Software Services through computing units at the BSs of the RAN.

The idea of placing the control over the network and resources in the cloudlets is based on *Sinaeepourfard et al.* [16]. The authors introduced a concept called *Integrated and Intelligent Control and Monitoring of IoT (I2CM-IoT)*. This idea places the control of the *cloudlet, Fog*, and *Edge* devices and their corresponding services inside the *cloudlet layer*. The main reasons behind this proposal are the following:

- The *cloudlets* are located inside the city, between the *Cloud* and the edge of the Smart City network. This makes *cloudlets* a potential bridge between the citizens and the centralized *Cloud*.
- The *cloudlets* can provide Services to the citizens while respecting local city policies and data privacy laws such as the General Data Protection Regulation (GDPR). The GDPR is explained in Section 2.4.1.2.
- As *cloudlets* are small data centers, they offer quite a lot of computing power. This makes the *cloudlet layer* suitable for executing services that do not require as much computing power as demanding *Cloud* services while keeping the data inside the city. Resulting in less bandwidth towards the *Cloud*, lower latency, and better data privacy for the citizens.

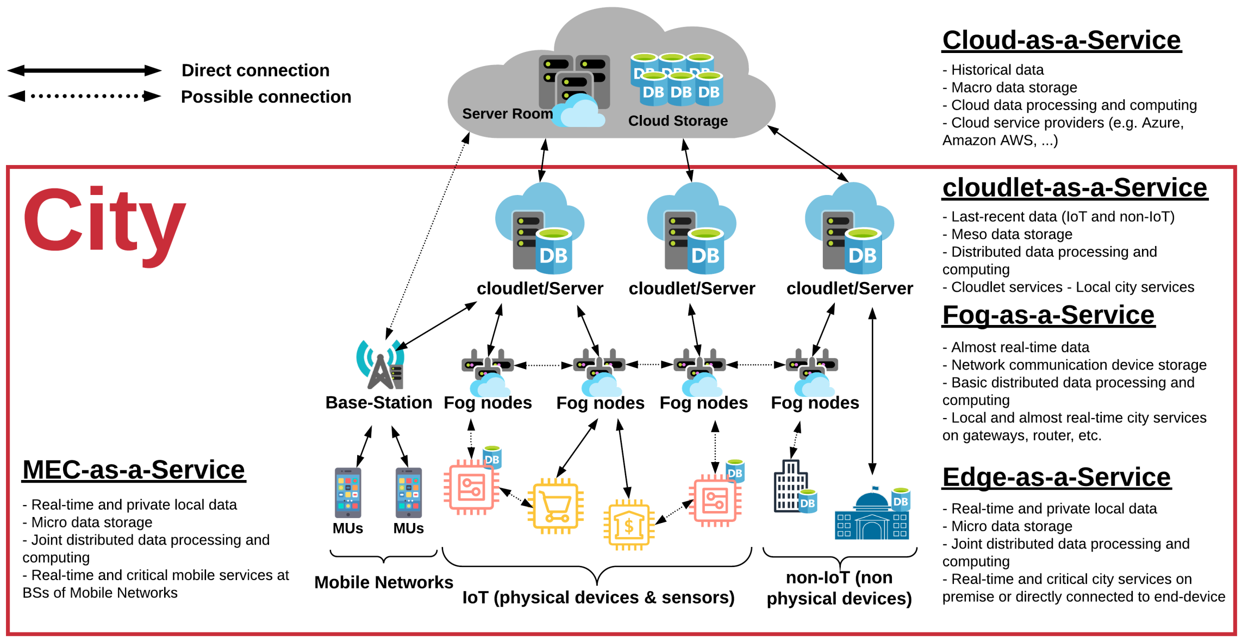

Direct connection
Possible connection
Server Room
Cloud Storage
DB DB DB
Cloud-as-a-Service
- Historical data
- Macro data storage
- Cloud data processing and computing
- Cloud service providers (e.g. Azure, Amazon AWS, ...)
City
cloudlet/Server
cloudlet/Server
cloudlet/Server
cloudlet-as-a-Service
- Last-recent data (IoT and non-IoT)
- Meso data storage
- Distributed data processing and computing
- Cloudlet services - Local city services
Fog-as-a-Service
- Almost real-time data
- Network communication device storage
- Basic distributed data processing and computing
- Local and almost real-time city services on gateways, router, etc.
Base-Station
Fog nodes
Fog nodes
Fog nodes
Fog nodes
MEC-as-a-Service
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical mobile services at BSs of Mobile Networks
MUs
MUs
Mobile Networks
IoT (physical devices & sensors)
non-IoT (non physical devices)
Edge-as-a-Service
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical city services on premise or directly connected to end-device

Figure 2.3: Software Service models in Smart Cities (Appendix A)

2.4 Developing Software Services in Smart Cities

This section explains a model on how to develop Software Services in Smart Cities. The selected papers from the literature review (Section 2.2) are classified based on this model.

In [17], *Javadzadeh et al.* present a taxonomy on applications in Smart Cities at the *Fog layer* based on a systematic literature review. This taxonomy serves as a base-line for the model of this thesis and can be seen in Fig. 2.4. The taxonomy is used to classify current work on *FC* in Smart Cities. The three main categories are *Service Objective*, *Application Classification*, and *Outcome Type*. The first category is based on the Service Objective. The authors have selected some common service requirements such as *Bandwidth Management, Latency Management, and Mobility* for this classification. The second category aims to categorize the work based on their application domain (e.g. Smart Healthcare, Smart Building, Smart Education, etc.). Finally, the authors make a distinction between the expected outcome type of the application. This can either be a platform, an architecture, or a framework.

The extended model, proposed in this study, contains four main categories. These categories can be seen as steps when developing Software Services. First of all, one has to determine the (non-)functional requirements of the service based on the goal and domain of the service. This step is called the *Classification of City Services*. The second step, the *Design Output*, is to design the services itself by selecting the appropriate technologies and architectures. This step pays attention to the *ICT Management Requirements*. These requirements are derived from the *City and ICT Objectives* from the previous step and determine the technologies that are used for developing the service. Afterward, the service must be developed and deployed. This step often results in a back-end service and a front-end service for the citizens to interact with. Finally, the *Efficiency Measurements* are taken based on the ICT Key Performance Indicators (KPIs) and the City/Use-case KPIs. These KPIs are often related to the *City Domain* and *City and ICT Objectives* from step 1. These four steps will be discussed in detail in the remainder of this section. The model can be found in Fig. 2.5.

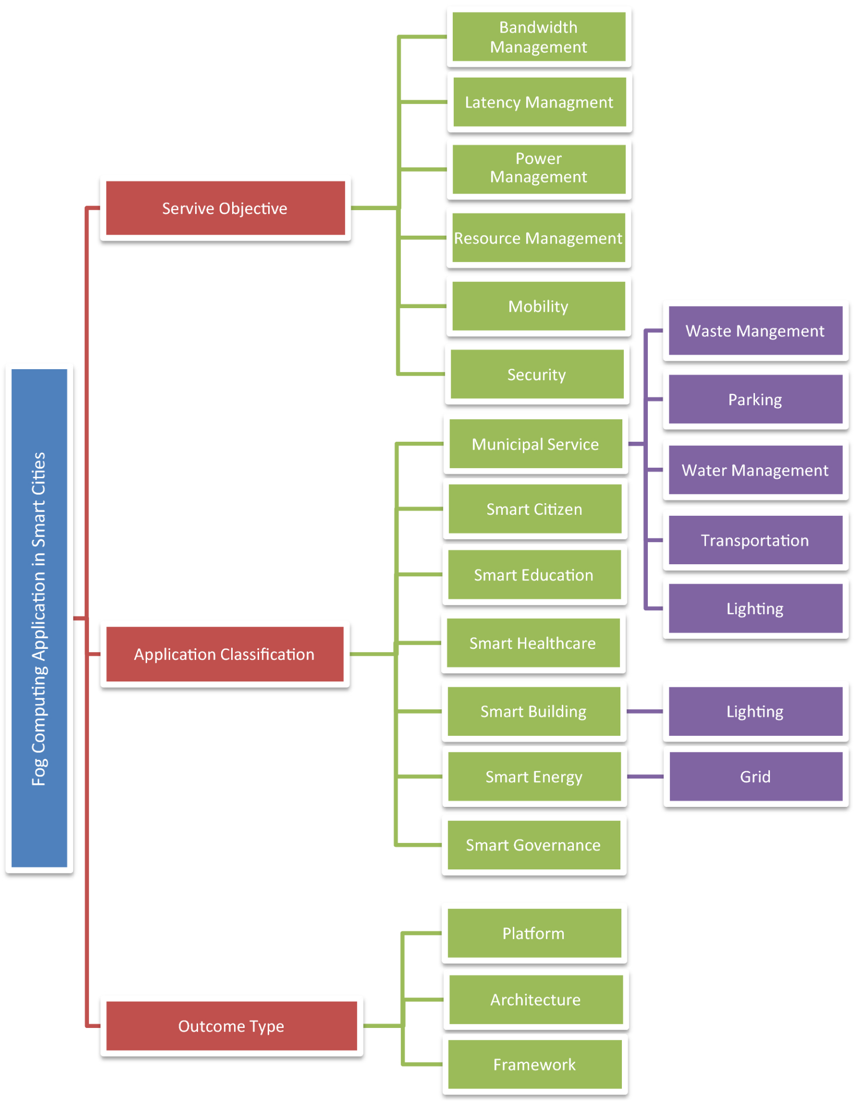


Figure 2.4: Taxonomy of Fog Computing applications in Smart Cities [17]

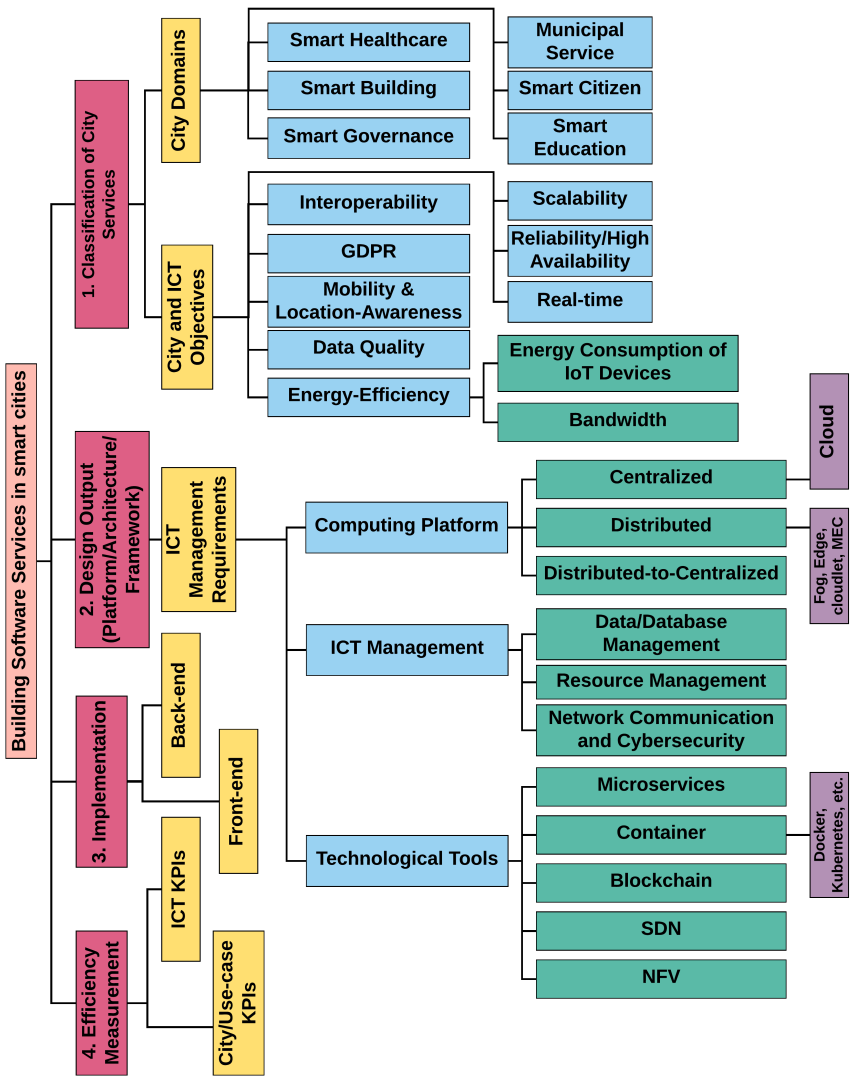


Figure 2.5: How to develop Software Services in Smart Cities

2.4.1 Classification of City Services

Before a service can be developed, one has to determine the functional and the non-functional requirements of the service. The functional requirements define what the service does or must not do. These requirements depend on the purpose of the service and will not be further discussed in this study. The non-functional requirements, however, depend more on the demand of the customers/citizens and the domain of the service, showing the importance of determining the *City Domain*. An emergency Smart Healthcare service, for example, will have different requirements than a Smart Education application for students.

The following studies are examples of proposals related to the Smart Neighborhood domain inside a Smart City. More information on Smart Neighborhoods can be found in Section 3.1.2.

Yang et al. [18] combines street lighting with sensing devices to create smart street lighting in Smart Neighborhoods. This proposal uses a container-based system for fast deployment and high scalability. Data is managed using NoSQL and in-memory databases. The data originating from the sensors can be processed using *EC*. Afterward, historical data is saved on Cloud servers This all while maintaining a secure transmission using an asymmetric key and Secure Shell (SSH) encrypted tunnel. *Zeng et al.* [19] proposes a security architecture schema based on Blockchain technology for existing smart traffic light systems. Another study by *Atif et al.* [20] is also related to smart technology in Smart Neighborhoods. They propose a system that improves the utilization of parking lots. This proposal uses predictive analytics to predict the transformation of parking spots in a parking.

The remainder of this section will discuss some common non-functional requirements for services in Smart Cities.

2.4.1.1 Interoperability

The term *Interoperability* in Softwares Services means that different computing and networking entities from the different Smart City layers must be able to work together in an efficient matter. This is an important requirement to consider because often when developing a new service and installing new hardware or software, some technology is already present. It is crucial in such a scenario to make sure that the old and new technologies can operate together.

2.4.1.2 GDPR

The General Data Protection Regulation (GDPR) is a privacy regulation for the protection of data brought into life by the European Union (EU). The GDPR on its own is not a non-functional requirement but its regulations oblige companies and organizations which handle data of consumers and citizens of Smart Cities in the EU to take specific requirements related to privacy and data security into account [21].

Badii et al. [22] present a Smart City architecture, "Snap4City", mainly focusing on the GDPR. The security platform is *Cloud*-based, but it also supports on-premise *Edge* applications which can connect to the platform using IoT brokers. The "Snap4City" architecture has been tested across a variety of test cases and scenarios. These extensive tests prove that their proposal is highly secure and satisfies the GDPR. It does seem like citizens who connect to a service of the platform in a secure way are obliged to go through the *Cloud* as all the security and privacy management is located in the *Cloud*. Additionally, a study of *Varadi et al.* [23] discusses the legal issues regarding *Fog, Edge, Cloud* and Artificial Intelligence (AI) applications related to the GDPR guidelines. The authors conclude their study with some recommendations for legal compliance of these applications.

2.4.1.3 Mobility and Location-Awareness

When a service or application has to support *Mobility and Location-Awareness*, it must be able to handle citizen/device movement from one area in the city to another, and possibly from one Smart City layer to another. Common examples of devices that can move to different locations and thus require support for *Mobility and Location-Awareness* are smartphones and drones.

Wei et al. [24] introduce a mobility-aware service caching system for *MEC* applications. The system tries to predict the target locations of the mobile users to forwards the service request to the appropriate BS. Tests show that the system significantly reduces service response time and increases the amount of services that are offered locally. Mobile devices also cause problems in the mobile health domain. *Dai et al.* [25] propose a computation offloading mechanism for mobile health applications. As applications are often too demanding for one mobile device, tasks are divided between nearby helpers (e.g. other emergency vehicles). The offloading system uses a Particle Swarm Optimization Algorithm to optimize the computation offloading problem. Simulations show that the proposed solution can efficiently decrease the task completion time of emergency healthcare applications.

2.4.1.4 Real-time

Many healthcare services are services that require fast response times and analysis. These kinds of services must be able to handle incoming data in real-time to satisfy the demands. This low-latency and real-time requirement is often necessary in critical applications.

Almeida et al. [26] present a real-time system for assisted living for elderly in a Smart City. Real-time data from sensors is processed on *Edge devices* using machine learning algorithms. This real-time data processing enables the system to alert medics and family members in case of emergencies. These notifications are sent using the MQTT protocol as a publish/subscribe communication protocol. More information on the MQTT protocol can be found in Section 2.4.5.6.

2.4.1.5 Scalability

A *Scalable* service must be able to adjust to a varying load and must support easy expansion when the amount of tasks increases.

As mentioned at the beginning of this section, *Yang et al.* [18] propose a smart street lighting system that uses a container technology for fast deployment and high scalability. The use of Docker containers and Docker Swarm enables the system to dynamically adjust its resources depending on the demand. More information on Container Technologies can be found in Section 2.4.5.2.

2.4.1.6 Reliability/High Availability

The service must be able to withstand node failures and must undergo as little downtime as possible.

2.4.1.7 Energy Efficiency

Energy Efficiency of Smart City services can be addressed in multiple ways. One possible kind of improving energy efficiency is through bandwidth management inside the city network. A different method for reducing the energy consumption is through an improved load scheduling system. Many algorithms for task scheduling that try to reduce energy consumption and evenly divide the load over the computing nodes are already extant.

Luo et al. [27] introduce a novel method for optimizing the runtime performance of continuous data flow applications. The reliability of nodes is verified using an anomaly detection method, as anomalous nodes can cause an increase in energy consumption and task delay. A block coordinate descend-based max flow algorithm is proposed to solve the latency-aware energy consumption optimization problem. Simulations show that the proposed method successfully optimizes the performance of continuous data flow applications. *Goudarzi et al.* [28] present an application placement technique in *Fog* and *Edge* IoT applications to minimize the execution time and energy consumption. A Memetic Algorithm is used for batch application placement. Additionally, the authors propose a pre-scheduling algorithm to maximize the number of parallel tasks that can be executed. Testing results based on their weighted cost model show that their proposal improves execution time and energy consumption of concurrent IoT applications.

2.4.2 Design Output

The goal of the second step is to design either an Architecture, a Platform, or a Framework [17]. This development process has to be executed while applying the ICT Management requirements related to the use case. These requirements often depend on the City and ICT Objectives as they determine what kind of platform, management strategies, and technological tools will be used.

The remainder of this section explains the difference between an Architecture, a Platform, and a Framework. Afterwards the different possibilities regarding the *Computing Platform* are discussed in Section 2.4.3. Next, the three main aspects of the *ICT Management* for Software Services in Smart Cities are examined. These aspects are *Data/Database Management*, *Resource Management*, and *Network Communication Management and Cybersecurity*. Finally, some common technological tools that were encountered during the literature review are given in Section 2.4.5.

- An architecture is an abstract blueprint or template on how a software system is designed and implemented. It is in fact the design of a structure, how the components of a system move and communicate [29].
- A platform can be perceived as a collection of technologies that are used as a foundation for other applications, processes, or technologies [30].
- A framework is a kind of architecture that is specifically designed to be extended, and to make the implementation of certain technologies easier [17, 31].

2.4.3 Computing Platform

There are three basic types of *Computing Platforms* in a Smart City scenario: “Centralized”, “Distributed”, and “Decentralized” platforms. The most common *Computing Platform* is a centralized computing platform. This kind of platform often uses cloud-based technologies (Section 2.1.2). All the storage and computing of the city data takes place inside the *Cloud*. Distributed computing platforms divide their computing power across multiple nodes are managed/owned by a larger entity. In a decentralized computing platform, all the nodes are connected to other nodes without any hierarchical structure. No master node has control over the other nodes. *cloudlets*, which are small datacenters that are often owned by different companies or organizations inside the City, can connect to form a decentralized computing platform (Section 2.1.3). However, they can also be part of a distributed platform when the *cloudlets* are connected to multiple *Fog Nodes* while acting as leader nodes over these *Fog Nodes*. Fig. 2.6 gives a visual representation of the different types of computing platforms.

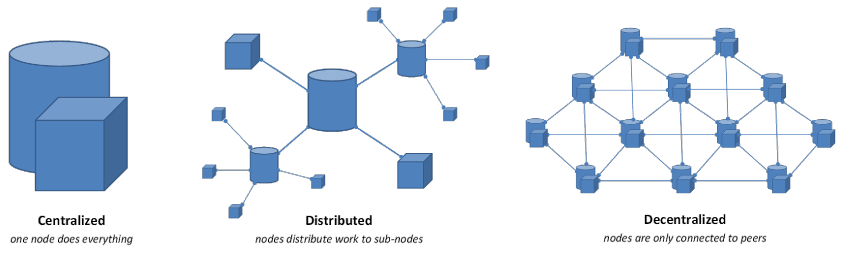

Centralized
one node does everything
Distributed
nodes distribute work to sub-nodes
Decentralized
nodes are only connected to peers

Figure 2.6: Centralized vs Distributed vs Decentralized [32]

As there are many different types of computing nodes and service providers inside a Smart City, the *Computing Platform* for Software Services is not limited to these scenarios. Another type of *Computing Platform* is a "Distributed-to-Centralized (D2C)" computing platform. This platform can be created by combining a centralized platform with one or more distributed computing platforms. *Sinaeepourfard et al.* [12, 33, 34] propose a Fog-to-Cloud data management architecture for Smart Cities. The goal of this architecture is to enable data access not only at the centralized *Cloud*, but also closer to the citizens at the *Fog Nodes*. In [16, 35, 36], the authors extend this idea by adding a *cloudlet layer* to the architecture and also addressing the *ICT Management* and *Software Management*.

2.4.4 ICT Management

One of the most important aspects of Smart Cities is the management of all its ICT resources. Current literature [16] mentions that there are three main categories when classifying the *ICT Management* tasks: *Data/Database Management*, *Resource Management*, and *Network Communication Management and Cybersecurity*. The remainder of this section will explain each of these management aspects and will give some specific examples related to each category.

2.4.4.1 Data/Database Management

The first *ICT Management* category is *Data/Database Management*. As the term implies, this step determines the data management methods and the database management methods.

According to *Sinaeepourfard et al.* [36], there are three main data types inside a Smart City: Historical Data, Last-recent Data, and Real-time Data. The Historical Data is located in the *Cloud*. This data is accessible from everywhere but with a relatively high latency. The Last-recent Data is located in the *cloudlet layer*. This data is accessible inside the city, closer to the citizens, thus with faster response times. Finally, the Real-time data is available at the edge of the network, at the sources of the data. The *E2CaaS* architecture (Fig. 2.3) also covers *EC*. Consequently, a fourth data type is added: Almost Real-time Data. The Real-time Data shifts to the *Edge Nodes*, as they can offer real-time services because they are almost directly connected to the IoT sources. Depending on the location of the *Fog Nodes*, they either house Real-time Data or Almost Real-time Data.

The management of these data types include every step concerning the lifecycle of the data in a Smart City. This lifecycle includes *Data Acquistition*, *Data Processing*, and *Data Preservation* [12]. The subtasks of each Part of the Data Lifecycle for a Smart City are shown in Fig. 2.7. It is important when designing a Smart City Software Service that these tasks are addressed when designing the data flow and selecting the tools for the service.

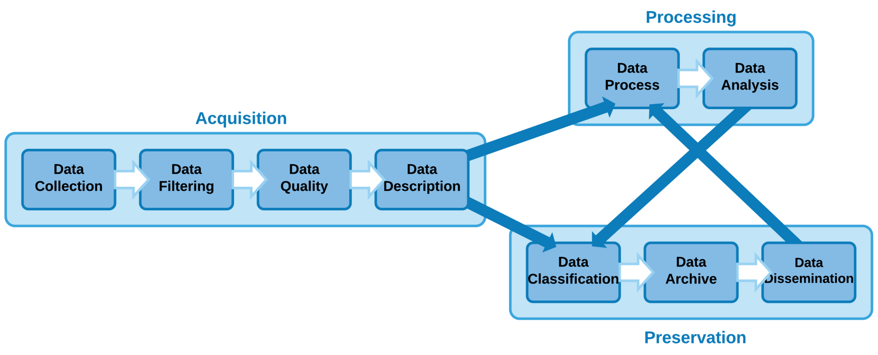


Figure 2.7: Comprehensive Scenario Agnostic Data Lifecycle model for a Smart City [12]

A second important aspect when handling data in a service is the database management. Selecting the right type of database for the acquired data is crucial. Currently, there are many options to choose from with each their corresponding strengths and weaknesses. Traditional databases are SQL-based databases. Lately, new types of databases, called NoSQL databases, are gaining popularity.

Yang et al. [18] use a NoSQL and an in-memory database to achieve flexible data management in their smart street lighting system.

Another notable trend that was encountered during the literature review related to data management is caching. The concept of data caching has been around for a while. Recently, it has also found its way into Smart Cities to reduce the data retrieval delay [13]. The following studies propose caching methods for Software Services in Smart Cities.

Xu et al. [37] present a Blockchain-based *Edge* caching scheme for mobile systems. To lower the content delivery delay to the end-user, the data is cached on the *Edge Nodes*. The Blockchain technology is used to tackle security and caching capacity issues on the *Edge Layer*. The Blockchain supervises the caching transactions between the mobile users and the *Edge Nodes*, while protecting the cache content. A max-min based algorithm is used to optimize the caching resources based on user demands. Simulations show that the proposed methods improve the utilities of the *Edge Nodes* and the Quality of Experience of the mobile users. As previously mentioned, *Wei et al.* [24] introduce a mobility-aware service caching system for MEC applications. The authors propose a service cache algorithm based on a back-propagation neural network. This Machine Learning algorithm results in the most popular services being cached on the *MEC Nodes*.

2.4.4.2 Resource Management

The second category is the *Resource Management*, involving the efficient organization of numerous ICT devices inside a Smart City. The main tasks of *Resource Management* are Resource Discovery, Resource Provisioning, Resource Scheduling, Offloading, and Load Balancing [13]. Resource Discovery is the process of identifying and locating existing resources inside the city. This process is mandatory for enabling the orchestrator to match the best suitable resources to its appointed services. The process of matching these resources to the services is called Resource Provisioning. This process provides parts of their allocated resources to the services. In other words, the orchestrator has some available resources which it has to divide between its services. Resource Provisioning is related to the Task Scheduling, Offloading, and Load Balancing processes. Task Scheduling divides the service in multiple subtasks and assigns these tasks individually to the available resources. When there are not enough resources available, the orchestrator can decide to offload the tasks to other orchestrators. For example, when a smartphone does not have the computing power for a task, the task can be offloaded to a MEC server. Finally, it is important to use a good Load Balancing method to make sure that not all the tasks are being scheduled onto the same group of computing nodes. Load Balancing can be improved by monitoring the load on the resources, which is called Resource Monitoring.

Wu et al. propose a task offloading scheme in a Vehicular Fog Network. The task offloading problem is based on semi-Markov decision process, which is solved by an iterative algorithm. This solution aims to reduce the transmission and computation delay of the tasks. Another study by *Sharmin et al.* [38], also related to the Internet of Vehicles, introduces a fog-federation environment where Smart Vehicles can either execute tasks themselves or offload the tasks to nearby *Fog Nodes*. The authors have designed a pricing model to offload the tasks. *Talaat et al.* [39] present a load balancing strategy using reinforcement learning for IoT Fog systems. The system continuously monitors the load on every server, the traffic in the network, and handles and distributes the requests/tasks. *Deng et al.* [40] again formulate the resource allocation process as a Markov Decision Process in an *EC* scenario. This time, the problem is solved using reinforcement learning. A policy is trained which generates resource allocation schemes that maximize the trustworthiness of the services.

2.4.4.3 Network Communication Management and Cybersecurity

The final ICT Management category addresses the communication between the ICT components in the city and the security of this communication. Tools like Software-Defined Networking (SDN) and Network Function Virtualization (NFV) can be used to control the access to and from the network resources. Both SDN and NFV will be explained in Section 2.4.5. Many other tools can be used for securing the communication between the network resources as these nodes often send confidential or sensitive data to each other.

Pham et al. [41] introduce a hierarchy of *Edge-to-Cloud* publish/subscribe broker overlay networks. An algorithm is designed to cluster these brokers based on topic similarities and location. This approach should reduce network traffic among brokers. Simulations show that the proposed method achieves optimal data delivery latency. The publish/subscribe paradigm is widely used because of its ability to loosely couple the entities in the network in terms of time, space, and synchronization.

To tackle the security problems of *EC* and *CC*, *Tian et al.* [42] propose a web attack detection system based on deep learning, which is deployed on *IoT Edge devices*. Experiments show that the system performs well when detecting web attacks. Due to the distributed technologies present in a Smart City, the network is often shared by many SPs who do not co-operate. *Ling et al.* [43] use a Blockchain-based RAN architecture to answer the complexities that come with authentication while the network is being shared by these SPs. The architecture can establish trust among SPs and to efficiently manage resources across the network.

2.4.5 Technological Tools

The third and final topic related to the *ICT Management Requirements* for the design of Software Services in Smart Cities is about the Technological Tools. These tools are used to address the tasks highlighted in the previous section (2.4.4) about the three categories of *ICT Management*. This section will go over some of the common tools that are used for Software Services in Smart Cities.

2.4.5.1 Microservices

The concept of Microservices architectures is to divide a service into multiple smaller services which are limited to one or a couple subtasks. The main advantages of these individual independent microservices over traditional monolithic architectures are improved scalability, improved speed, and more flexibility in terms of which technologies can be used [44]. The downside to these microservice architectures, according to [45], is that it requires more design and planning. It also needs good load balancing algorithms to keep its efficiency up. Additionally, microservices are more prone to cyber-attacks due to the independent services which have to trust each other for their communication. The difference between a monolithic architecture and a microservices architecture is visualized in Fig. 2.8.

Zhou et al. [46] proposes an approximation latency-aware microservice mashup

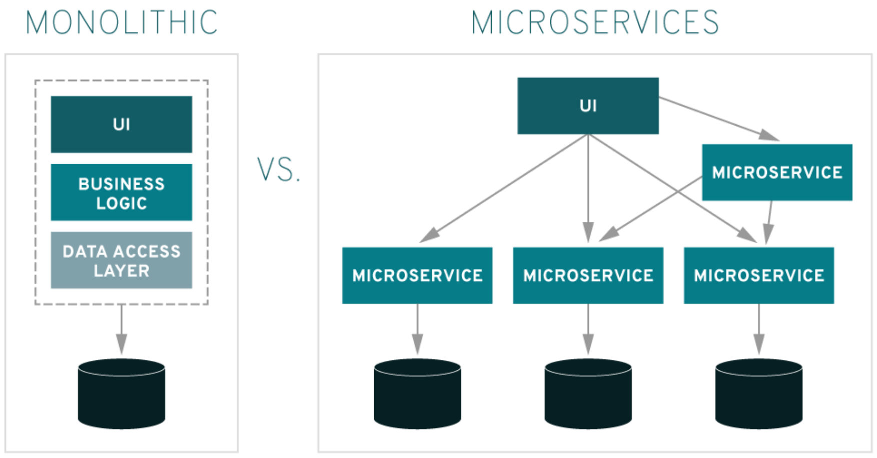


Figure 2.8: Monolithic architecture vs Microservices architecture [44]

approach to solve the problem of collocating IoT application microservices in *MEC*. This problem is an integer non-linear programming problem which is NP-hard. Microservices are used because they are easy to deploy through Container Technologies and offer more flexibility compared to monolithic applications.

2.4.5.2 Container Technologies

According to [47], container technologies are lightweight virtualization technologies which enable the deployment and execution of large-scale distributed applications on *Cloud, cloudlet, Edge*, and *Fog* platforms. The main advantages of container technologies are their easy life-cycle management, negligible overhead while running, reduced start, restart, and stop time compared to Virtual Machines (VMs), and portability. The three main components of container technologies are the containers itself, the container manager, and the container orchestrator. Containers come in various shapes and sizes, depending on the product of choice, e.g. Docker, LXC, OpenVZ, etc. A container is a unit of software that puts the code and the dependencies of an application/service inside a package to make the application quickly and reliably deployable across computing environments [48]. Fig. 2.9 shows the difference between a Docker-based application and a VM-based application. It is clearly visible that the containerized applications share a single OS, where the VMs each have their own OS. This makes them more lightweight and decrease their start, restart, and stop times by a lot. The lifecycle of the containers can be managed through an Application Programming Interface (API), exposed by the container manager. It is the task of the container orchestrator to manage the deployment, monitoring, and configuration of multi-container applications. Examples of well-known orchestrators are Docker Swarm and Kubernetes.

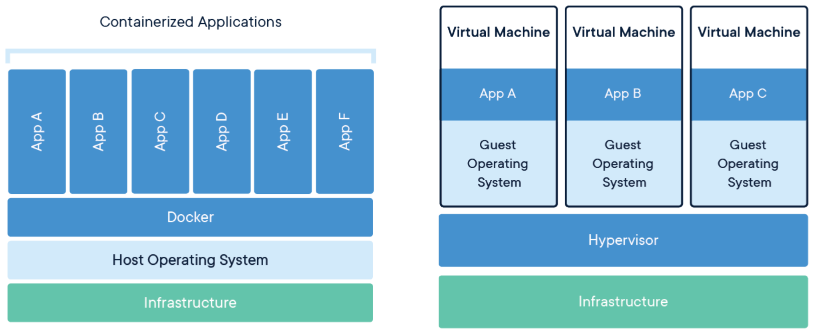


Figure 2.9: Container-based application vs VM-based application [48]

As mentioned above, *Yang et al.* [18] use Docker containers and Docker Swarm for their smart street lighting system. The authors chose containerization because of its high scalability when deploying *Cloud* and *Edge* services and because of the good balance between flexibility and performance overhead of Docker containers. They compared different features of Docker containers and VMs in Table 2.5 below.

Table 2.5: Comparison of Docker and Virtual Machines [18]

Features	Docker	Virtual Machine
Start time:	$<$ 50ms	30 - 45 seconds
Stop time:	$<$ 50ms	5 - 10 seconds
System overhead:	No overhead	Overhead
Storage space:	Tens of MBs in size	Tens of GBs in size
Scalability:	Highly scalable	Not easily scalable
CPU load when idle:	Normal	$\pm$ 1.5% more than docker
Isolation:	Less isolation	More isolation
Network round trip time:	$\pm$ 75 μ s	$\pm$ 60 μ s
Throughput (read and write):	10 000 I/Os per second	5000 I/Os per second

2.4.5.3 Blockchain

Blockchain is a technology that is recently getting a lot of attention from a variety of industries including information technologies such as IoT, CC, and Big Data. The authors of [49] explain Blockchain as data and information blocks which are chained together chronologically. The blocks are safely recorded by encryption. Blockchain uses a distributed node consensus algorithms to generate and update blocks in the chain. In other words, a node cannot add data blocks to the chain unless the data gets validated by enough other nodes. The main reasons why Blockchain is a worthy candidate for providing secure network communication between nodes in distributed IoT environments is its consistency, integrity of data transmission, transparency, and distributed nature [50].

Xiao et al. [51] propose an IoT architecture "EdgeABC" for *Edge* applications that uses Blockchain technology to ensure the integrity of transaction data and profits. The authors also propose a Task Offloading and Resource Allocation algorithm which is built upon the benefits of the Blockchain. A study by *Aujla et al.* [52] proposes a Blockchain-based secure data processing framework for an edge Vehicle-to-X environment. Blockchain is used to target the following problems: 1. potential inappropriate utilization of *Edge Nodes*, 2. Operational challenges in data integrity and caching, and 3. High energy consumption due to link reestablishment after link breakage. Blockchain provides consistency and integrity of data during migration from one node to another. Plus, it can be used for pre-caching.

2.4.5.4 SDN

Software-Defined Networking (SDN) is a networking technique where the data and control plane are separated by creating a virtualized control plane that manages the network functions. By doing so, SDN bridges the gap between service provisioning and network management. It enables the network to be directly programmable using southbound interfaces (e.g. "OpFlex", "OpenFlow", etc.), resulting in a flexible network that can adapt to dynamic situations like changing network conditions, and changing needs of businesses, markets, and citizens [53].

In a study by *Baktir et al.* [54], SDN is described as an enabler for technologies like *cloudlet Computing, FC, EC,* and *MEC*. By simplifying the network management, defining network flows, increasing network capability, and facilitating virtualization, SDN can decrease the complexity of these technologies. The authors present a collaboration model to show the feasibility of their ideas. The architecture of the model can be seen in Fig. 2.10. The architecture uses “OpenFlow” southbound interfaces to connect the *Edge Layer* to northbound applications (i.e. Service Orchestrator, Load Balancer, etc.). The southbound interfaces form the forwarding plane, while the northbound applications are the programmable control plane of the network. This control plane is accessed by the SDN Controller through the northbound API.

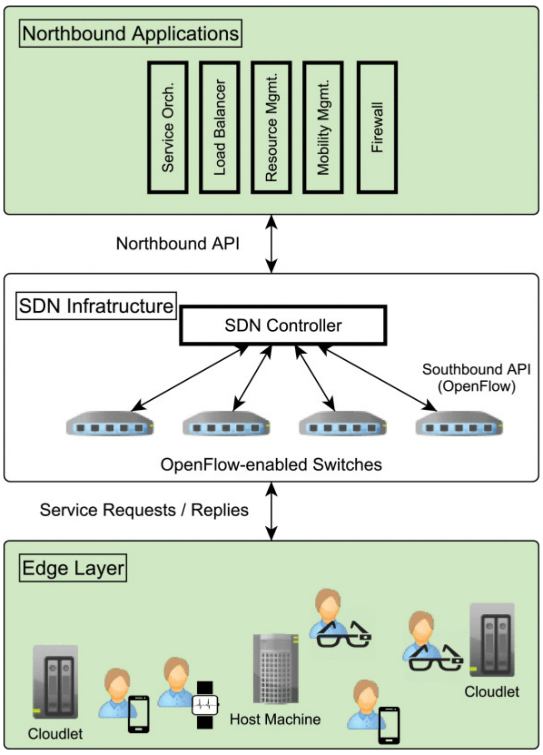


Figure 2.10: Collaboration model between SDN and distributed technologies [54]

2.4.5.5 NFV

According to [53], Network Function Virtualization (NFV) is the virtualization of network functions on top of commodity servers. This enables the SP to use commodity hardware instead of dedicated hardware devices for these network functions, thus greatly reducing the initial and operational cost. SPs are also able to easily add, remove, or update functions for a specific subset of customers. Additionally, NFV offers flexibility to scale services up or down based on customer demands, making it more practical to offer tailored services to customers.

Mouradian et al. [55] present a novel PaaS architecture for a *Fog/Cloud*-based system. As this architecture is use-case driven, the authors describe two different scenarios. These two applications are decomposed into multiple functions. These Virtual Network Functions (VNFs) are chained together in a forwarding graph. A placement algorithm is developed for deploying this graph and its VNFs onto the available resources. The system is also able to discover existing resources. The authors chose NFV because it enables them to deploy network functions as software instances on general-purpose hardware. *Xu et al.* [56] propose a placement strategy for placing VNFs in a MEC environment. Simulations show that the proposed algorithm outperforms other algorithms significantly. This study also chose NFV because it lowers the operational cost of the network functions by replacing the dedicated hardware with software running in VMs.

2.4.5.6 MQTT

Message Queuing Telemetry Transport (MQTT)[3] is an open, lightweight, easy-to-implement, client-server publish/subscribe messaging transport protocol. Because of its characteristic, MQTT is popular messaging protocol in IoT and Machine-to-Machine communications. MQTT uses a message broker, the MQTT Broker. The MQTT Broker receives messages from clients and routes them to other clients. Messages are published on *topics*. Other clients can subscribe to these topics to receive the published messages. The authors chose this protocol because it is lightweight and thus can run on IoT devices with low computing capabilities. Plus, it enables the devices to send messages to the broker asynchronously.

Almeida et al. [26] uses MQTT in their real-time medical assistance application for elderly. Data from IoT sensors in the houses of the elderly are processed using *EC*. In case of an emergency, notifications are sent to family members or medics

[3]`https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html`

using the MQTT protocol.

2.4.5.7 Machine Learning

Machine Learning is an application of AI. It allows a system to learn and improve based on experience. Authors in [57] state that Machine Learning has advanced a lot lately due to the growing interest in IoT and sensing technologies. Large quantities of data are being collected and stored that can be used for Machine Learning applications. Advancements in Graphics Processing Unit (GPU) technologies also aid the rapid evolution of Machine Learning. Finally, there are many advanced Machine Learning algorithms available, improving the efficiency of Machine Learning applications. Machine Learning can be used for a diversity of applications in Smart Cities such as Resource Scheduling, data anomaly detection, and intrusion detection.

Wei et al. [24] use a backpropagation neural network for their *Edge* caching system. The neural network is used to predict the most popular services. These services are then cached on the *Edge Nodes*. A neural network is a set of algorithms that are designed to recognize patterns. They are especially useful for clustering and classifying. Additionally, *Farhat et al.* [58] propose a solution for an on-the-fly deployment of *Fog Nodes* near users. The deployment and scheduling of these nodes must be done cleverly because the demand can change randomly. In order to solve this scheduling problem, the authors propose a reinforcement R-learning model. This model can produce a *Fog* placement schedule based on the behavior of clients. R-learning is a variant of Q-learning, which is a model-free reinforcement learning algorithm. The main feature of Q-learning is that does not require a model of the environment. R-learning uses Q-learning based on Average Reward. This reward is a combination of rewards earned during the transition between states and should be maximized. Experiments show that the proposed solution can reduce the amount of requests towards the *Cloud* and thereby reducing its load. It is also notable that the services are deployed using the Kubernetes orchestrator and Docker containers.

2.4.6 Implementation

The third step is implementing the selected technologies and the service logic from the previous steps. Most Software Services consist of a Front-End and a Back-End. The Back-End contains most of the logic and algorithms used for the service. It regulates how the applications work. The Front-End is used by the citizens to access the logic inside the Back-End. The interaction between the Back-End and the Front-End is often through an API or a Software Development Kit (SDK). Fig. 2.11 by *Tönjes et al.* shows how the Front-End and Back-End can be connected through an API in a scenario where IoT and social media stream-based applications are integrated.

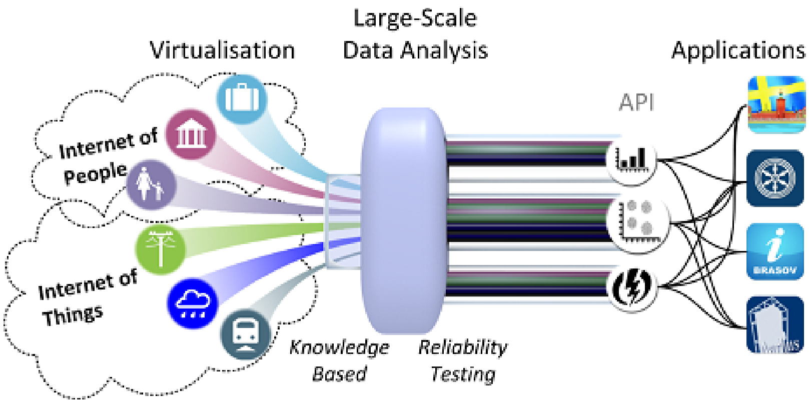


Figure 2.11: Integration of IoT and social media stream-based applications [59]

2.4.7 Efficiency Measurements

The final step for building software Services in Smart Cities is to take and analyze the *Efficiency Measurements*. These measurements are taken in the shape of KPIs. A KPI is a measurable value that indicates how well the service achieves key business objectives. They can be obtained by performing simulations, measurements, and surveys on the service [36]. Two types of KPIs should be considered: ICT KPIs and city/use-case KPIs. ICT KPIs [16] are general measurable values like bandwidth and latency. They are related to the *Data/Database Management*, the *Resource Management*, and the *Network Communication Management and Cybersecurity*. City/Use-case KPIs depend on the requirements of the Smart City or the specific domain/use-case of the service. The use-case of this study is discussed in Chapter 3.

3 EMS Software Services in Smart Neighborhoods

As Chapter 2 discussed a general overview of developing Software Services in Smart Cities, this chapter goes narrower into the specific use-case of this study. First of all, some background information regarding the use-case on the ZEN Research center, Smart Neighborhoods, Smart Grid, and EMSs is given. Afterward, *Q2.1* is solved by applying the proposed model from the previous chapter to this scenario. The final section of this chapter gives some directions towards the implemented in Chapter 4, which addresses *Q2.2*.

3.1 Background

3.1.1 ZEN research center

The ZEN Research Center researches Zero Emission Neighborhoods (ZEN) in Smart Cities. The goal of the ZEN center is to contribute to a low carbon society by achieving no emission of greenhouse gas in the neighborhoods of Smart Cities[1]. The research center tries to achieve this goal by developing solutions for future buildings and neighborhoods. Their research is conducted through eight pilots in Norway. These pilots are located in Bodø, Trondheim, Steinkjer, Evenstad, Elverum, Oslo, Bærum, and Bergen [12].

One of the interests of the ZEN Center is to develop an architecture for building efficient Software Services for Smart Cities and Smart Neighborhoods (*Q2.1*). This architecture can be used to develop Energy Management System (EMS) services that can reduce the energy consumption and improve the Quality of Life in the city and its neighborhoods.

3.1.2 Smart Neighborhoods

Smart Cities consist out of Smart Neighborhoods. By focusing on the neighborhoods of the city instead of the city itself, some complexities that come with large scale EMSs can be eliminated. Plus, by keeping the data and the services as local and close the citizens as possible, more respect to privacy and data security can be paid.

3.1.3 Smart Grid

The electrical grid in a city or neighborhood is a network connecting electrical energy producers to consumers. Components of this grid are generating stations, transmission stations, transmission lines, and distribution lines [60]. The generation stations generate electrical energy (i.e. power plants). This energy can be stepped up or down in the transmission stations and is transported and delivered to the consumers via the transmission lines. A *Smart Grid*, indicates

[1]`https://fmezen.no/`

that technologies, ranging from meters to advanced software are used to manage and improve the functionalities of the electrical grid. These technologies give the grid the characteristics of a computer network. The main characteristics of such a *Smart Grid* are listed below [60].

- Reliability: Due to automated grid analysis and fault detection, problems in the grid are detected more quickly. Resulting in faster solutions or even avoidance.
- Flexibility: New generating plants can be connected to the existing grid in a plug-and-play fashion, providing more flexibility to the grid. Additionally, there is more support for renewable energy sources such as solar, wind, and water plants.
- Efficiency: Due to, among others, Demand-Side Management the efficiency of the energy infrastructure can be improved. Demand-Side Management (DSM) is a concept that tries to reduce the amount of energy consumed during peak hours [61]. This does not reduce the total energy consumption but it releases some of the pressure from the power plants. DSM also opens new opportunities for renewable energy sources such as solar energy plants, because these kinds of energy plants are not able to satisfy the otherwise high peak demands. Challenges like load balancing and Peak-to-Average ratio reduction [62] are important factors that can improve the grid's efficiency.

3.1.4 Energy Management Systems

EMSs are systems in Smart Homes, Smart Neighborhoods, or Smart Cities that control the energy consumption. They are often connected to the *Smart Grid* for monitoring and managing energy consumption. Plus, they enable citizens to understand and interact with their energy consumption [63]. An EMS can for example guide inhabitants of a Smart Home through the process of DSM to avoid energy consumption during peak hours. Fig. 3.1 by *Aman et al.* [63] illustrates the role of an EMS in a Smart Grid.

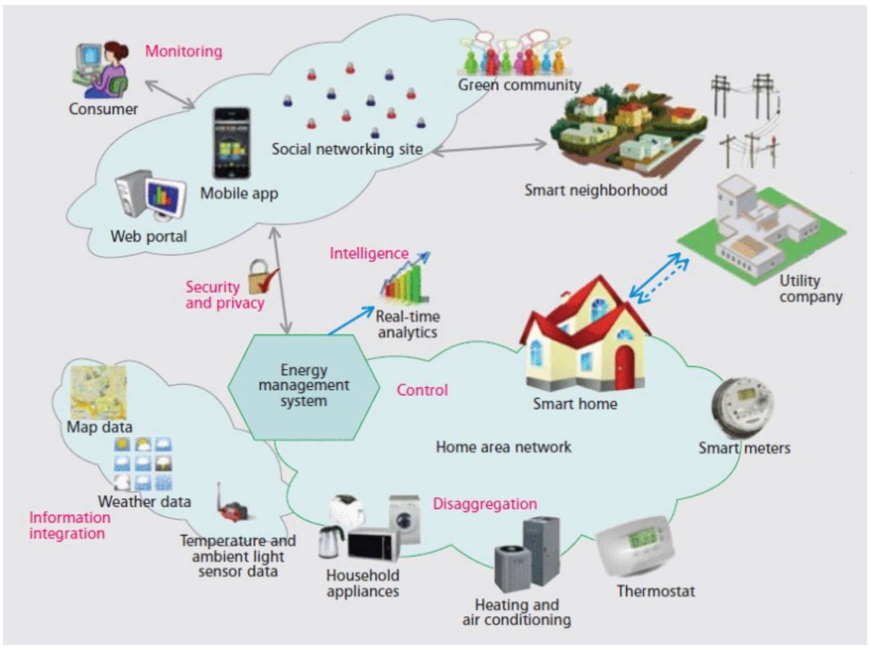


Figure 3.1: Energy Management System in a Smart Grid [63]

3.2 Edge-to-Cloud-as-a-Service architecture for the ZEN Center

After gaining some background knowledge on the ZEN Center, EMS, etc., the first step towards a solution for *Q2.1* can be taken. In the previous chapter, an *E2CaaS* architecture is proposed for Software Services in Smart Cities. The goal of this chapter is to adapt this architecture to the ZEN Center, and to tailor the model on how to develop Software Services in Smart Cities to this architecture. The resulting *E2CaaS* for the ZEN Research Center is shown in Fig. 3.2.

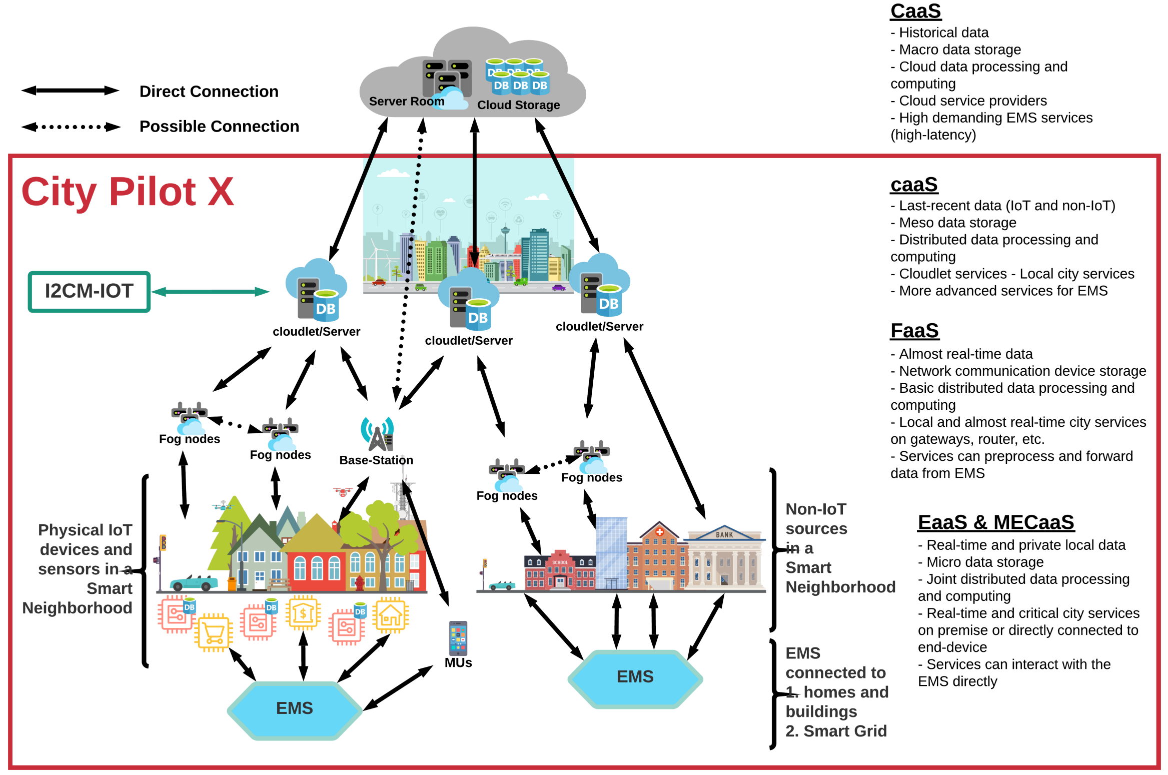

Direct Connection
Possible Connection
Server Room
Cloud Storage
DB DB DB
CaaS
- Historical data
- Macro data storage
- Cloud data processing and computing
- Cloud service providers
- High demanding EMS services (high-latency)
City Pilot X
I2CM-IOT
DB
cloudlet/Server
cloudlet/Server
cloudlet/Server
caaS
- Last-recent data (IoT and non-IoT)
- Meso data storage
- Distributed data processing and computing
- Cloudlet services - Local city services
- More advanced services for EMS
FaaS
- Almost real-time data
- Network communication device storage
- Basic distributed data processing and computing
- Local and almost real-time city services on gateways, router, etc.
- Services can preprocess and forward data from EMS
Fog nodes
Fog nodes
Base-Station
Fog nodes
Fog nodes
Physical IoT devices and sensors in a Smart Neighborhood
Non-IoT sources in a Smart Neighborhood
EaaS & MECaaS
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical city services on premise or directly connected to end-device
- Services can interact with the EMS directly
MUs
EMS
EMS
EMS connected to
1. homes and buildings
2. Smart Grid

Figure 3.2: E2CaaS architecture for ZEN Center

Firstly, one can see the IoT devices located inside the Smart Homes and on various places inside the Smart Neighborhoods. In a EMS scenario, these devices are sensors, smart meters, etc. that are connected to the *Smart Grid* via the EMS. These devices generate data necessary for the Software Services in the Smart Neighborhoods. For private or critical real-time applications, the citizens can make an appeal to the *Edge nodes* that are located close by. These nodes can be dedicated computing devices or small local datacenters that do not need a connection to the Internet. The IoT devices can also send their data to the *Fog nodes* inside the Smart Homes, as these nodes are often gateways or switches. The *Fog nodes* can execute small tasks that do not require a lot of computing power such as data preprocessing. They can also offload more demanding tasks such as training a machine learning model, to the *cloudlets* in the Smart City or Neighborhood. These *cloudlets* have decent computing capabilities and are well connected to the Internet. When services have to be applied to multiple pilots (cities) at once, the communication will go through the *Cloud*. With almost unlimited computing capabilities, the *Cloud* can execute the most demanding tasks. Additionally, Fig. 3.2 shows that MEC can have its place in a Smart Neighborhood. The citizens can use their mobile devices, such as smartphones, to monitor and manage the EMS of their Smart Home.

The non-IoT datasources are located on the bottom right. This non-IoT data is produced by human-human or human-machine interactions. The companies and organizations that produce this kind of data can have their own *cloudlet* at their disposal. Consequently, the datasources in the figure are directly connected to the *cloudlet layer*. If this is not the case, some *Fog nodes* can be used to processes or forward the data. The data can then be used for services, or can simply be stored on a *cloudlet* or on the *Cloud*.

After executing the tasks for a specific service, the computing entities in the Smart City and its neighborhoods can report back and manage the utilization of the *Smart Grid* in the neighborhoods through the EMSs.

3.3 Developing EMS Software Services in Smart Neighborhoods

This section tries to answer *Q2.1* by looking at the model of Chapter 2 in Fig 2.5 and tailoring this model to the ZEN research center and our E2CaaS architecture from the previous section. Fig. 3.3 shows the result. Like Chapter 2, this figure will be explained step by step. Alongside this, references towards the role of the steps inside the E2CaaS architecture will be clarified.

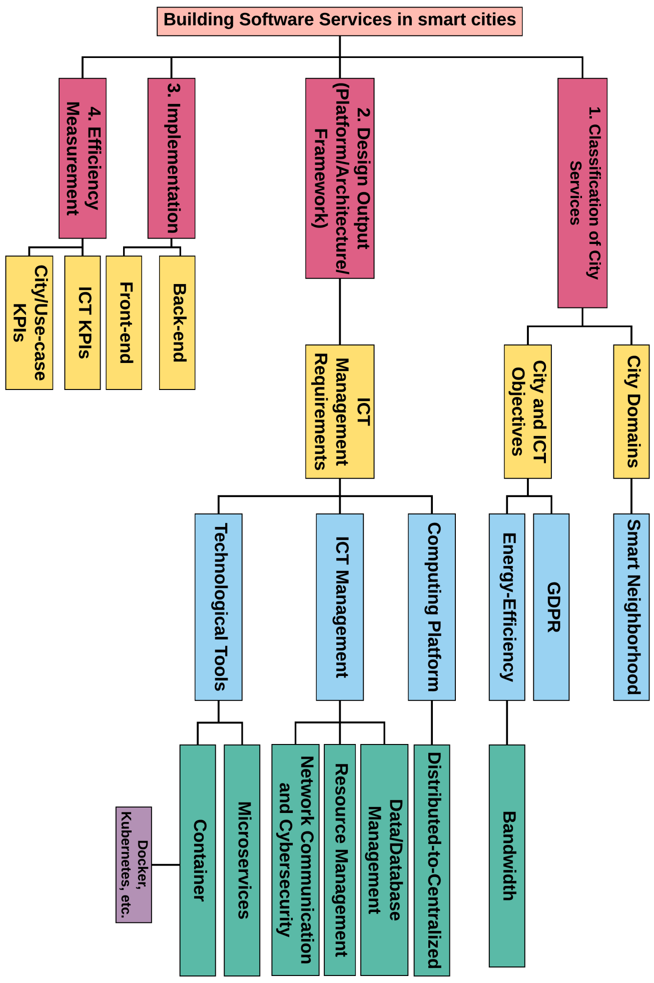


Figure 3.3: Developing EMS Software Services for Smart Neighborhoods

3.3.1 Classification of City Services

Section 3.1.1 explained how the ZEN Research Center is connected to the Smart Neighborhoods. It is important to the research center to develop software solutions that contribute to the Zero Emission Neighborhoods (ZEN)s. In Fig. 3.3, the Smart Neighborhood and EMS domains are selected. These domains automatically bring some objectives along. Firstly, the services are located close to the citizens in the neighborhoods of a Smart City and will probably process private data from these citizens. This leads to the first City and ICT Objective: the GDPR. As mentioned in Section 2.4.1.2, when a company or organization processes or stores data from inhabitants of the EU, they must comply with the GDPR. This involves paying special attention to the security and the privacy of the citizens' data. Concepts such as the "Right to be forgotten" must be implemented [21]. Secondly, the goal of the services for the ZEN Center is to enable or to aid zero-emission of greenhouse gasses. This means that another important objective of the services can be the energy efficiency. Addressing the bandwidth used by new or existing services can lead to improved energy efficiency and thus reduce the amount of greenhouse gas emission.

3.3.2 Design Output

When addressing the ICT Management of the services, not only a solution for the *Data/Database Management* but also for *Resource Management* and the *Network Management and Cybersecurity* must be developed. The literature review from Chapter 2 indicates that there are many different approaches and technologies available for these problems. Most of these solutions are useful for specific services and scenarios, as they all have their strengths and weaknesses. This section discusses the *Computing Platform* and *ICT Management* categories. The *Technological Tools* are discussed later on in Section 3.4 when giving directions towards the implementation.

3.3.2.1 Computing Platform

ZEN Center is mainly interested in a *Distributed-to-Centralized* architecture. This type of *Computing Platform* enables both privacy-friendly and real-time computation at the edge of the Smart City, and computing-intensive applications in the *Cloud*. As the research center is also working with pilots spread across Norway, there are possibilities for computing applications to run on individual pilots, but also for applications that run in the *Cloud* and cover multiple pilots.

Fig. 3.4 shows how the distributed and centralized computing platforms can be placed inside the pilots. Close to the citizens, in the Smart Homes and Smart Buildings, there are possibilities for *EC* and *FC*. The *Fog nodes* can connect to the dedicated local datacenters (*cloudlets*) to offload tasks and send data. Mobile users can connect to the computing units at the BSs, which are connected to the local datacenters. The different pilots are connected through a centralized *Cloud*.

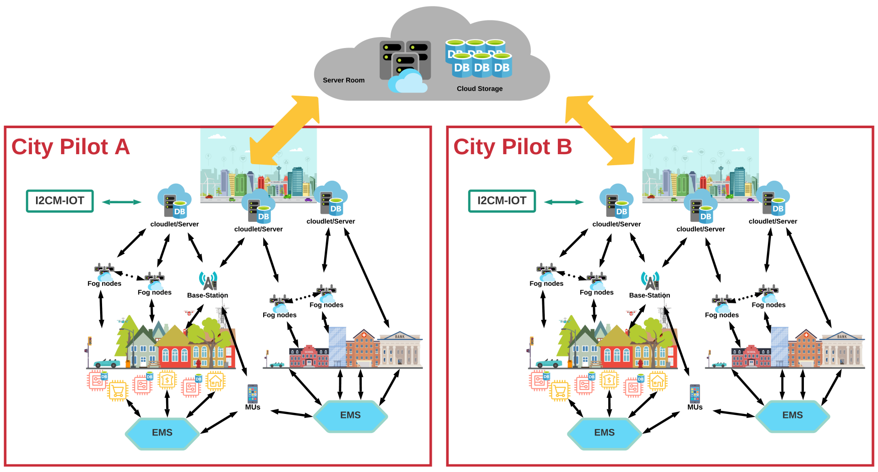


Figure 3.4: Connecting multiple pilots through the E2CaaS architecture

3.3.2.2 Data/Database Management

In terms of database selection, there are many options to choose from. As the type(s) of database that should be used is/are highly dependent on the specific service that will be implemented, there is no general suggestion for this architecture. However, when giving directions towards the implementation of a service in Section 3.4, an example of the selection of the database is explained.

Sinaeepourfard et al. [36] has already defined a data management architecture for the ZEN Center scenario. Three main data types have been identified: Context

Data, Research Data, and KPI Data[2].

- Context Data is the data coming from IoT devices or external sources. This data can be processed and interpreted but is not useful by itself. Examples are temperature data, sensor data from energy systems, and weather data.
- Research Data is generated by special dedicated applications such as simulations or data planning applications. This data comes from live buildings in the pilots. This can be for example citizen behavior or energy data.
- KPI Data is gathered based on the predefined KPIs of the ZEN Center. This type of data is collected by carrying out surveys, simulations, and measurements. KPI data is mainly used in the fourth and final step of building software services: *Efficiency Measurements*.

Fig. 3.5 shows where these data types are generated and are accessible in a Smart City. The Context Data is available in the Micro, the Meso, and the Macro level [36]. This data can be collected for each building, for each neighborhood, or for the city as a whole. Next, the Research data can exist on the Meso and Macro level. Finally, the KPI data can exist again on all three levels. Furthermore, it also shows how these data types are related to the Smart City data types. On the Macro level, the Context, Research, and KPI data is stored as Historical data. This means that the data was not produced right now and can be used later on. The Meso level can contain Last-recent data and Almost Real-time data from the Micro level. As mentioned, there can be KPI data at the Meso level when simulations, surveys, and measurements are carried out. This data can come in the form of either Almost Real-time or last-recent data. On the Micro level, the context data produced by various physical devices in the neighborhood can be produced as Real-time data. The Micro level can also produce Real-time KPI data close to the end-users.

3.3.2.3 Resource Management

The *Resource Management* for the use-case is similar to that of the general model for Smart Cities. In the proposed architecture from Fig. 3.2, the orchestration is located in the *cloudlet layer* using the proposed I2CM-IoT method of

[2]`https://fmezen.no/`

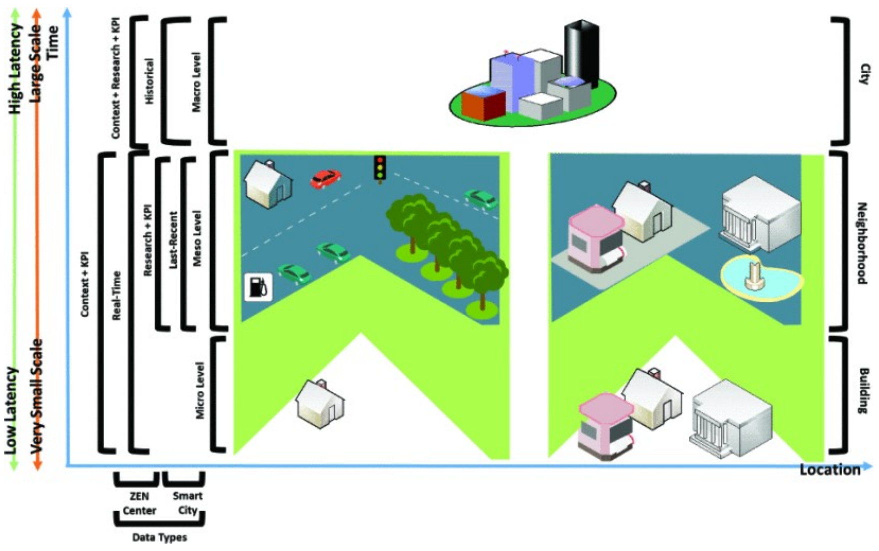


Figure 3.5: Data types in ZEN Center [36]

Sinaeepourfard et al. [16]. A first possible solution could use a container orchestrator located on the *cloudlets*. The orchestrator can then proceed to manage the resources inside the neighborhoods of the Smart City. The disadvantage of centralized orchestrators like Kubernetes is that they need their master node to be reliable and highly available. This results in most systems placing the orchestrator in the *Cloud layer*. A possibility that should be explore in future work is to divide the city in multiple clusters. These smaller clusters could then be managed from the *cloudlet layer* and potentially enable this concept. Another approach could utilize SDN controllers for managing the resources. However, as with the container orchestrators, the programmable control plane of SDN is centralized. Consequently, a solution for distributing this control plane across the *cloudlet nodes* must be designed.

3.3.2.4 Network Communication Management and Cybersecurity

As with the selection of the database, there are many available technologies for the network communication. The right technology or protocol highly depends on the purpose of the service, the preferences of the developers, and the type of data that is sent between nodes. For real-time services, messaging queue technologies such as Apache Kafka Streams[3] (based on the Kafka Messaging Protocol) or one of the various MQTT Brokers (Section 2.4.5.6 for more information on MQTT) are a good option as they enable asynchronous communication between nodes and high scalability. Nodes can also subscribe to specific topics, simplifying the work of the sender. Simple applications that do not require these characteristics can use HTTP Requests or other protocols such as REST[4].

Security and privacy between the computing nodes can be accomplished through the same technologies as for a general Smart City service. Section 2.4.4.3 has discussed some possible technologies such as Blockchain and SDN. For securing the EMS of Smart Homes and Smart Neighborhoods, other proposals have been made. For example, a study by *Mugarza et al.* [64] proposes a solution for security in EMS Smart City applications using the Cetratus framework [65] for enabling Dynamic Software Updates (DSUs).

3.3.3 Implementation

Like the *Design Output*, the implementation depends on the purpose of the service that will be developed. There are many frameworks and tools available in a variety of programming languages for both Front-End and the Back-End.

A Front-End for managing the ZEN KPIs is already introduced by *Sinaeepourfard et al.* in [16].

3.3.4 Efficiency Measurements

The ZEN Center has defined seven categories with sets of assessment criteria and KPIs for achieving zero emission neighborhoods. These seven categories are Greenhouse Gas Emission, Energy, Power/Load, Mobility, Economy, Spatial Qualities, and Innovation. *Sinaeepourfard et al.* [16] place these criteria and KPIs inside a “ZEN KPI box.” This box is then connected to the “ZEN Toolbox,” presented in [66]. These two boxes work together with the I2CM-IoT control entity in the *cloudlet layer* to measure the performance and asses the Software Services of a specific pilot.

[3]`https://kafka.apache.org/documentation/streams/`
[4]`https://restfulapi.net/`

3.4 Directions towards implementation

This section will give directions towards the implementation of a simplified EMS Software Service in Chapter 4, used for solving *Q2.2*. First of all, a quick sketch of the scenario is given. Afterward, the selected *Technological Tools* and the motivation behind this selection are discussed.

3.4.1 Scenario and Goals

The service that will be implemented uses the proposed E2CaaS architecture for the ZEN center. The service reads temperature values from sensors of an EMS in a Smart Home. The purpose of this service is to execute a simple processing task on the incoming temperature values. This should enable the service to send feedback or alerts to the citizens and the EMS of their home.

To goal of this implementation, is to answer *Q2.2* by validating the proposed E2CaaS for ZEN architecture and by giving an example on how to build a Software Service according to the model from Fig. 3.3. The main focus in terms of technology lies on the use of containerization. Additionally, some different setups of the service are implemented to compare the load on the resources in a distributed and centralized layout.

3.4.2 Technological Tools

This section explains the selection of *Technological Tools* for managing the ICT components.

3.4.2.1 Tools for Data/Database Management

The data that is processed and sent to other nodes in the service are temperature values. As these values are simple floats they do not require special NoSQL databases such as MongoDB[5] that allow for document-like database entries.

[5] `https://www.mongodb.com/`

Possible lightweight databases are PostgreSQL[6], MySQL[7], and SQLite[8].

3.4.2.2 Tools for Resource Management

The main focus of the implementation, in terms of technology, lies on containerization. The containerization product of choice for this implementation is Docker[9]. Docker is one of the most popular containerization platforms that offers easily deployable and manageable containers. Plus, it works well together with the chosen Container Orchestrator: Kubernetes[10]. Kubernetes is also widely used and one of the most popular container orchestrators at the moment of writing. It enables developers to automate the deployment, scaling, and management of containerized applications, making it an excellent choice for managing the resources.

3.4.2.3 Tools for Network Communication Management and Cybersecurity

Kubernetes is compatible with a lot of plugins that enable network communication between cluster nodes. The networking plugin that was selected for this implementation is the Weave Net plugin[11]. This plugin is selected because it is easy and quick to deploy on the cluster nodes and does not require any configuration. However, the plugin offers barely any privacy and security features out of the box. There are no other measurements taken for the security and privacy in this implementation. Consequently, the cybersecurity aspect should be further explored in future work.

The data is sent from node to node using HTTP POST requests. This method is widely used and easy to set up. HTTP Requests are chosen for the communication because the service in this scenario does not require any asynchronous communication and is executed on a small scale. Future Work should consider diving into the possibilities of messaging queue publish/subscribe-based alternatives such as the MQTT protocol instead.

[6]https://www.postgresql.org/
[7]https://www.mysql.com/
[8]https://www.sqlite.org/index.html
[9]https://www.docker.com/
[10]https://kubernetes.io/
[11]https://www.weave.works/oss/net/

4 Implementation

This chapter goes into detail about the setup and results of the implementation that is introduced in Section 3.4. Firstly, the scenario is highlighted. Next, the working of the used products and technologies is explained. Finally, the implementation is evaluated based on the setup experience and some measurements.

4.1 Scenario and Setup

As mentioned before, the goal of this implementation is that validate the proposed architecture from the previous chapter and to give a simplified example of how EMS Software Services can be developed. The Software Service that is described, is based on a temperature sensor from an EMS in a Smart Home, Smart Building, or Smart Neighborhood. Data is collected at the edge of the network and has to be sent upwards in the direction of the *Cloud* for data processing. In this scenario, a basic processing job is deployed on the *cloudlet layer*. The result of this processing job is reduced anonymous data that can be sent towards the *Cloud layer*. This data can be saved as Historical data inside this layer and can be used for various applications. The *cloudlet* receiving and processing the data can use the incoming Last-recent data to give feedback to the EMS or to generate notifications when, for example, a specific threshold is met. Four variants of this setup are implemented. Afterward, they are compared in Section 4.2 in terms of CPU usage of the *Cloud nodes* and *cloudlet nodes*, and in terms of the amount of requests the IoT devices can send to the upper layers. Fig. 4.1 illustrates these different setups. All these implementations are deployed on the imec IDLab Virtual Wall[1]. The concept of the Virtual Wall is briefly explained in Section 4.1.1.

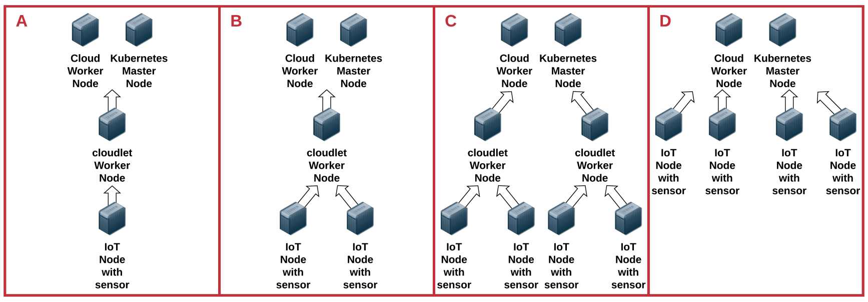


Figure 4.1: Different implementations of the temperature data processing service

The *Cloud layer* has two nodes: a Cloud Worker Node, and a Kubernetes Master Node. The Cloud Worker Node is used to handle incoming requests from the lower layers. The Kubernetes Master Node manages the Kubernetes cluster and allocates the resources. More detailed information on how Kubernetes works can be found in Section 4.1.2. The *cloudlet nodes* are part of the Kubernetes cluster and each house a container with the data processing service. Finally, the IoT nodes read temperature data from a dataset file and send this data to the upper layers using HTTP POST requests. The dataset used for this project was found on the IEEE Open Data Sets-webpage[2].

The results are obtained by sending the complete dataset from the IoT devices to the *cloudlets* or *Cloud* at an incrementing maximum rate (i.e. first test at maximum 1000 requests per second, second test at max 1500 requests per second, etc.). Meanwhile, the actual rate at which the data is being delivered and the load on the Central Processing Units (CPUs) of all the devices is measured. This

[1]`https://doc.ilabt.imec.be/ilabt/virtualwall/index.html`
[2]`https://site.ieee.org/pes-iss/data-sets/`

allows for the comparison of the load on the devices based on the specific scenarios.

4.1.1 Virtual Wall

The Virtual Wall is a testbed by IDLab, imec, and Ghent University for advanced networking, distributed software, cloud, big data, and scalability research and testing. Users can select and reserve nodes from a variety of different types of devices. Afterward, the reserved nodes can be set up however the user likes. Table 4.1 shows the specifications of the nodes used for this project.

Table 4.1: Specifications of the Virtual Wall nodes

Node Type	**CPU**	**RAM**
Cloud Worker Node	2x Quad core Intel E5520 (2.2GHz)	12GB
Kubernetes Master Node	2x Quad core Intel E5520 (2.2GHz)	12GB
cloudlet Worker Nodes	2x Quad core Intel E5520 (2.2GHz)	12GB
IoT Nodes	2x Dual core AMD opteron 2212 (2GHz)	8GB

4.1.2 Kubernetes

Kubernetes[3] is an open-source platform that enables easy deployment and management of containerized applications. Applications are split up into different parts. These parts are built into containers. Containers share the OS of the node that they are running on, making them lightweight and fast. Kubernetes can manage these different containers and ensure that there is no downtime. According to their website, the main features of Kubernetes are:

- Service Discovery and Load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing

[3]`https://kubernetes.io/`

- Secret and configuration management

These functionalities make Kubernetes an excellent choice for addressing many aspects of the *E2CaaS* architecture.

The downside is that Kubernetes is developed around the idea of *CC*. Consequently, the orchestration part of the cluster should run in a robust, centralized location. This makes it less viable for the proposed I2CM-IoT control box in the *cloudlet layer*. However, there are already some alternatives available such as KubeEdge[4] which are worth looking into in future work.

In terms of how the services was deployed using Kubernetes, firstly, a new Kubernetes cluster is initiated on the Kubernetes Master Node. The network plugin used for communicating inside the cluster is the Weave Net plugin[5]. Next, the services for the *Cloud Worker Node* and the *cloudlet Worker nodes* are placed inside a Docker container and pushed to Docker Hub[6]. Afterward, the worker nodes are added to the cluster. By labeling the nodes based on their layer, the containers can be deployed on their corresponding layers using a deployment. To make the *cloudlet nodes* accessible from the IoT Nodes (outside the cluster), the *cloudlet* deployment is exposed as a service with type *NodePort*, allowing the service to be reachable through an exposed port on the worker nodes. The *Cloud* deployment is also exposed internally to the cluster to enable it to receive requests from the *cloudlets*. Finally, the IoT Nodes can start sending requests to the *cloudlet layer*.

4.2 Results

The first graph in Fig. 4.2 shows that connecting multiple IoT devices to a single *cloudlet* or *Cloud node* decreases the amount of requests that the devices can send per second. However, if this number is multiplied by the number of nodes, the result is a much higher number of requests per second. In this setup, when connecting two IoT nodes, the individual performance of the IoT devices is reduced by 8%, and when connecting four devices, it is reduced by 19%. But

[4]`https://kubeedge.io/en/`
[5]`https://www.weave.works/`
[6]`https://hub.docker.com/`

when taking into account that there are now multiple devices sending data simultaneously, an overall increase of respectively 84% and 225% is observed.

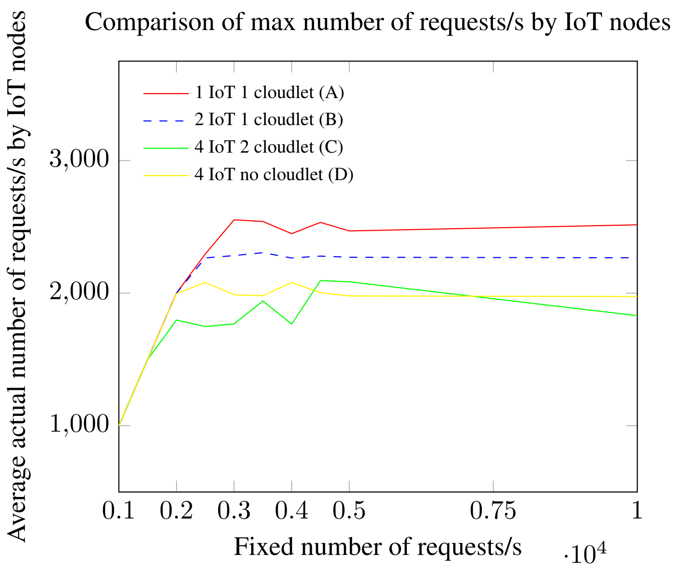


Figure 4.2: Maximum number of requests per second of IoT nodes

The second graph in Fig. 4.3 shows the average CPU usage of the IoT nodes in function of the maximum number of requests per second. As the graph illustrates, the highest CPU usage, at an average of 15.3%, occurs when four IoT nodes are sending requests to a single *Cloud node*. In the other scenarios, it seems that using two IoT devices and one *cloudlet* uses less CPU on the IoT devices (within 2%). This could be caused by inconsistencies on the IoT devices, as sometimes one of the IoT devices would consume a lot of CPU while the others remain at a more normal percentage. Nonetheless, using *cloudlets* allows the IoT devices to remain at a CPU usage of around 10-12%, while the *Cloud* scenario consumes around 15%. This can have a big impact on real-world IoT devices that do not have a lot of computing power.

The graph in Fig. 4.4 shows the average CPU usage of the *cloudlet nodes* when increasing the maximum requests per second on the IoT devices. Adding an extra IoT device to a *cloudlet* increases the load on the *cloudlet* by 93%. This is also to be expected as the device has to process twice as many requests than in scenario A when doubling the number of IoT devices.

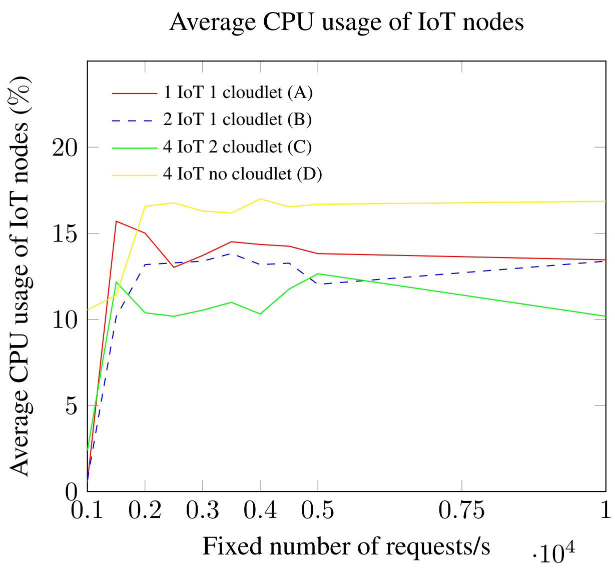


Figure 4.3: Average CPU usage of IoT nodes

Finally, Fig. 4.5 is more useful in terms of comparing the distributed approach from scenario C to the centralized layout from scenario D. The graph shows that moving the computational tasks to the *cloudlets* greatly reduces the load on the *Cloud*. However, with this setup, the results do become more variable. This is visible when looking at the green lines in the graphs. They are not as constant as the other lines because the performance of the *cloudlet* and IoT nodes fluctuated more than in the other scenarios.

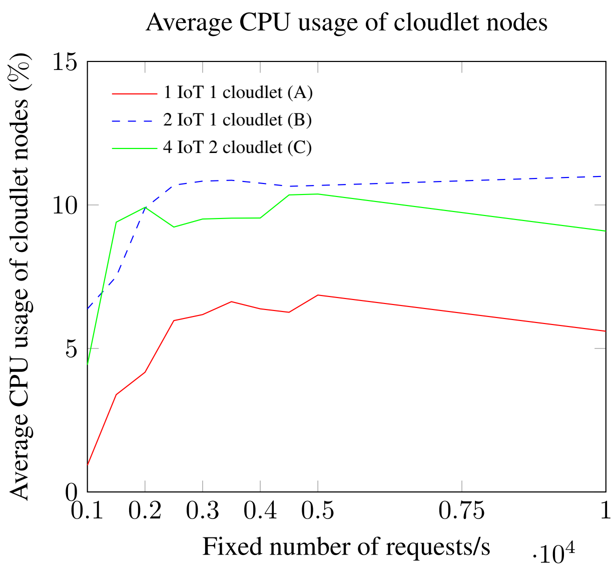


Figure 4.4: Average CPU usage of cloudlet nodes

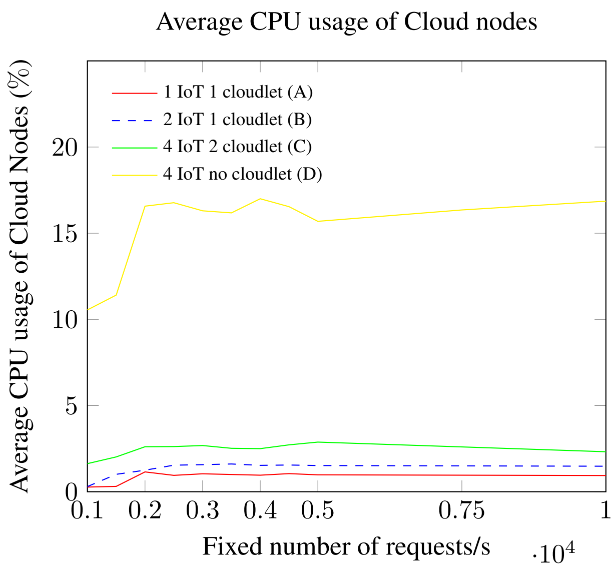


Figure 4.5: Average CPU usage of Cloud nodes

4.3 Discussion

The hardest part in this implementation, is setting op the Kubernetes cluster and configuring the devices in a persistent way (i.e. not having to manually reinstall everything when restarting the nodes). Once the cluster is up and running, the services are easily deployable thanks to Kubernetes and Docker. These tools also leave almost unlimited opportunities for improving and extending the system. For example, one can easily add a Front-end running on the *cloudlet nodes* for visualization of and interaction with the incoming data.

The testing results show that bringing the computing tasks closer towards the edge of the network can effectively reduce the amount of work of the *Cloud*. Additionally, by prepocessing the data and not sending all the data to the *Cloud*, less bandwidth towards the *Cloud* is used.

The author does believe that there are some limitations with using the HTTP methods for sending data. A publish/subscribe protocol such as MQTT would be a better alternative and will definitively improve the performance and scalability of the existing system. As mentioned before, it could be worth it to look into alternatives to Kubernetes such as KubeEdge to try to bring the control layer closer towards the edge of the network as proposed in the *E2CaaS* architecture. This was not possible in this implementation due to time constraints.

5 Conclusion and Future Work

In this Master's dissertation, a general solution on how to build Software Services in Smart Cities is designed and formulated. This goal was achieved by firstly carrying out a broad literature review on Software Services in Smart Cities based on distributed technologies. Using this knowledge, an *E2CaaS* architecture, which includes some of the most common distributed and centralized technologies, is proposed. Alongside this architecture, a model on how to develop Software Services in Smart Cities is presented. The purpose of this model is to guide the reader through the process of developing Software Services. The second part of the thesis applied this architecture and model onto the ZEN Research Center. The goal of this part of the dissertation is to explore the possibilities of developing Software Services for EMS and to show how the model from Chapter 2 can be applied onto a general use-case. Finally, to validate the proposed model and to explore the possibilities of containerization, a simplified Software Services that can be used for EMSs is implemented using Docker containers and the Kubernetes orchestrator. Four different scenarios are compared to visualize the impact of distributing the computing tasks over *cloudlets* instead of centralizing these tasks on the *Cloud*. The results show that moving the computing tasks to the *cloudlets* can reduce the datastream towards and the load on the *Cloud* significantly. Notably, this work does not aspire to give a complete overview of all the possible technologies and methods that are available to develop Software Services in Smart Cities. Alternatively, the author aims to give a general direction towards common tools and technologies that can enable the development of Software Services. This study servers as a foundation for future work on developing Software Services for Smart Cities and for the ZEN Research Center.

Future work should aim to improve the proposed architecture for the ZEN Center by moving the control layer to the *cloudlets*. Alternatives to the Kubernetes orchestrator, such as KubeEdge, should be explored when addressing this issue. Additionally, the possibilities and performance in terms of scalability when using a publish/subscribe communication protocol, such as MQTT, should be studied. A part of the ICT Management category in the model that was ignored for this study was the security and privacy aspect. With regulations getting more strict and more data of the citizens being used for services, this is an important topic that should definitely be researched in the future. Finally, a more complete implementation of a Software Service should be implemented using the proposed *E2CaaS* architecture based on real-world data from one of the pilots of the ZEN Center.

References

[1] "Internet of Things Global Standards Initiative," https://www.itu.int/en/ITU-T/gsi/iot/Pages/default.aspx.

[2] A. H. Alavi, P. Jiao, W. G. Buttlar, and N. Lajnef, "Internet of Things-enabled smart cities: State-of-the-art and future trends," *Measurement*, vol. 129, pp. 589–606, Dec. 2018.

[3] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: State-of-the-art and research challenges," *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, May 2010.

[4] "What is IaaS? Infrastructure as a Service — Microsoft Azure," https://azure.microsoft.com/en-us/overview/what-is-iaas/.

[5] A. G. Prajapati, S. J. Sharma, and V. S. Badgujar, "All About Cloud: A Systematic Survey," in *2018 International Conference on Smart City and Emerging Technology (ICSCET)*, Jan. 2018, pp. 1–6.

[6] "What is PaaS? Platform as a Service — Microsoft Azure," https://azure.microsoft.com/en-us/overview/what-is-paas/.

[7] "What is SaaS? Software as a Service — Microsoft Azure," https://azure.microsoft.com/en-us/overview/what-is-saas/.

[8] N. Agarwal and G. Agarwal, "Role of Cloud Computing in Development of Smart City," in *National Conference on Road Map for Smart Cities of Rajasthan (NC-RMSCR*, Apr. 2017.

[9] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The Case for VM-Based Cloudlets in Mobile Computing," *IEEE Pervasive Computing*, vol. 8, no. 4, pp. 14–23, Oct. 2009.

[10] R. Huang, Y. Sun, C. Huang, G. Zhao, and Y. Ma, “A Survey on Fog Computing,” in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, ser. Lecture Notes in Computer Science, G. Wang, J. Feng, M. Z. A. Bhuiyan, and R. Lu, Eds. Cham: Springer International Publishing, 2019, pp. 160–169.

[11] C. Mouradian, D. Naboulsi, S. Yangui, R. H. Glitho, M. J. Morrow, and P. A. Polakos, “A Comprehensive Survey on Fog Computing: State-of-the-Art and Research Challenges,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 416–464, Firstquarter 2018.

[12] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, “Data Preservation through Fog-to-Cloud (F2C) Data Management in Smart Cities,” in *2018 IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, May 2018, pp. 1–9.

[13] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, “All one needs to know about fog computing and related edge computing paradigms: A complete survey,” *Journal of Systems Architecture*, vol. 98, pp. 289–330, Sep. 2019.

[14] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, “On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, thirdquarter 2017.

[15] T. X. Tran, A. Hajisami, P. Pandey, and D. Pompili, “Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges,” *IEEE Communications Magazine*, vol. 55, no. 4, pp. 54–61, Apr. 2017.

[16] A. Sinaeepourfard, J. Krogstie, and S. A. Petersen, “A Distributed-to-Centralized Smart Technology Management (D2C-STM) model for Smart Cities: A Use Case in the Zero Emission Neighborhoods,” in *2019 IEEE International Smart Cities Conference (ISC2)*, Oct. 2019, pp. 760–765.

[17] G. Javadzadeh and A. M. Rahmani, “Fog Computing Applications in Smart Cities: A Systematic Survey,” *Wireless Networks*, vol. 26, no. 2, pp. 1433–1457, Feb. 2020.

[18] Y.-S. Yang, S.-H. Lee, G.-S. Chen, C.-S. Yang, Y.-M. Huang, and T.-W. Hou, “An Implementation of High Efficient Smart Street Light Management System for Smart City,” *IEEE Access*, vol. 8, pp. 38 568–38 585, 2020.

[19] P. Zeng, X. Wang, H. Li, F. Jiang, and R. Doss, “A Scheme of Intelligent Traffic Light System Based on Distributed Security Architecture of Blockchain Technology,” *IEEE Access*, vol. 8, pp. 33 644–33 657, 2020.

[20] Y. Atif, S. Kharrazi, D. Jianguo, and S. F. Andler, “Internet of Things data analytics for parking availability prediction and guidance,” *Transactions on Emerging Telecommunications Technologies*, vol. 31, no. 5, p. e3862, May 2020.

[21] “What is GDPR, the EU’s new data protection law?” https://gdpr.eu/what-is-gdpr/, Nov. 2018.

[22] C. Badii, P. Bellini, A. Difino, and P. Nesi, “Smart City IoT Platform Respecting GDPR Privacy and Security Aspects,” *IEEE Access*, vol. 8, pp. 23 601–23 623, 2020.

[23] S. Varadi, G. Gultekin Varkonyi, and A. Kertesz, “Legal Issues of Social IoT Services: The Effects of Using Clouds, Fogs and AI,” in *Toward Social Internet of Things (SIoT): Enabling Technologies, Architectures and Applications: Emerging Technologies for Connected and Smart Social Objects*, ser. Studies in Computational Intelligence, A. E. Hassanien, R. Bhatnagar, N. E. M. Khalifa, and M. H. N. Taha, Eds. Cham: Springer International Publishing, 2020, pp. 123–138.

[24] H. Wei, H. Luo, and Y. Sun, “Mobility-Aware Service Caching in Mobile Edge Computing for Internet of Things,” *Sensors*, vol. 20, no. 3, p. 610, Jan. 2020.

[25] S. Dai, M. Li Wang, Z. Gao, L. Huang, X. Du, and M. Guizani, “An Adaptive Computation Offloading Mechanism for Mobile Health Applications,” *IEEE Transactions on Vehicular Technology*, vol. 69, no. 1, pp. 998–1007, Jan. 2020.

[26] A. H. Almeida, I. Santos, J. Rodrigues, L. Frazão, J. Ribeiro, F. Silva, and A. Pereira, “Real-Time Low-Cost Active and Assisted Living for the Elderly,” in *Ambient Intelligence – Software and Applications –,10th International Symposium on Ambient Intelligence*, ser. Advances in

Intelligent Systems and Computing, P. Novais, J. Lloret, P. Chamoso, D. Carneiro, E. Navarro, and S. Omatu, Eds. Cham: Springer International Publishing, 2020, pp. 153–161.

[27] Y. Luo, W. Li, and S. Qiu, "Anomaly Detection Based Latency-Aware Energy Consumption Optimization For IoT Data-Flow Services," *Sensors (Basel, Switzerland)*, vol. 20, no. 1, Dec. 2019.

[28] M. Goudarzi, H. Wu, M. S. Palaniswami, and R. Buyya, "An Application Placement Technique for Concurrent IoT Applications in Edge and Fog Computing Environments," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[29] "ARCHITECTURE — meaning in the Cambridge English Dictionary," https://dictionary.cambridge.org/dictionary/english/architecture.

[30] "PLATFORM — meaning in the Cambridge English Dictionary," https://dictionary.cambridge.org/dictionary/english/platform.

[31] "FRAMEWORK — meaning in the Cambridge English Dictionary," https://dictionary.cambridge.org/dictionary/english/framework.

[32] "What is the Difference Between Centralized, Distributed and Decentralized AI Computing?" Oct. 2018.

[33] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Fog-to-Cloud (F2C) Data Management for Smart Cities," in *Saionference on FTC 2017*, Vancouver, Canada, Nov. 2017, p. 8.

[34] A. Sinaeepourfard, J. Krogstie, and S. A. Petersen, "D2C-DM: Distributed-to-Centralized Data Management for Smart Cities Based on Two Ongoing Case Studies," in *Intelligent Systems and Applications*, ser. Advances in Intelligent Systems and Computing, Y. Bi, R. Bhatia, and S. Kapoor, Eds. Cham: Springer International Publishing, 2020, pp. 619–632.

[35] A. Sinaeepourfard, S. A. Petersen, and D. Ahlers, "D2C-SM: Designing a Distributed-to-Centralized Software Management Architecture for Smart Cities," in *Digital Transformation for a Sustainable Society in the 21st Century*, ser. Lecture Notes in Computer Science, I. O. Pappas, P. Mikalef, Y. K. Dwivedi, L. Jaccheri, J. Krogstie, and M. Mäntymäki, Eds. Cham: Springer International Publishing, 2019, pp. 329–341.

[36] A. Sinaeepourfard, J. Krogstie, S. A. Petersen, and D. Ahlers, "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Apr. 2019, pp. 590–595.

[37] Q. Xu, Z. Su, and Q. Yang, "Blockchain-Based Trustworthy Edge Caching Scheme for Mobile Cyber-Physical System," *IEEE Internet of Things Journal*, vol. 7, no. 2, pp. 1098–1110, Feb. 2020.

[38] Z. Sharmin, A. W. Malik, A. Ur Rahman, and R. MD Noor, "Toward Sustainable Micro-Level Fog-Federated Load Sharing in Internet of Vehicles," *IEEE Internet of Things Journal*, vol. 7, no. 4, pp. 3614–3622, Apr. 2020.

[39] F. M. Talaat, M. S. Saraya, A. I. Saleh, H. A. Ali, and S. H. Ali, "A load balancing and optimization strategy (LBOS) using reinforcement learning in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, Feb. 2020.

[40] S. Deng, Z. Xiang, P. Zhao, J. Taheri, H. Gao, J. Yin, and A. Y. Zomaya, "Dynamical Resource Allocation in Edge for Trustable Internet-of-Things Systems: A Reinforcement Learning Method," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6103–6113, Sep. 2020.

[41] V.-N. Pham, V. Nguyen, T. D. T. Nguyen, and E.-N. Huh, "Efficient Edge-Cloud Publish/Subscribe Broker Overlay Networks to Support Latency-Sensitive Wide-Scale IoT Applications," *Symmetry*, vol. 12, no. 1, p. 3, Jan. 2020.

[42] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A Distributed Deep Learning System for Web Attack Detection on Edge Devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, Mar. 2020.

[43] X. Ling, Y. Le, J. Wang, and Z. Ding, "Hash Access: Trustworthy Grant-Free IoT Access Enabled by Blockchain Radio Access Networks," *IEEE Network*, vol. 34, no. 1, pp. 54–61, Jan. 2020.

[44] "What are microservices?" https://www.redhat.com/en/topics/microservices/what-are-microservices.

[45] A. V. Nene, C. T. Joseph, and K. Chandrasekaran, "Construing Microservice Architectures: State-of-the-Art Algorithms and Research

Issues," in *Knowledge Management in Organizations*, ser. Communications in Computer and Information Science, L. Uden, I.-H. Ting, and J. M. Corchado, Eds. Cham: Springer International Publishing, 2019, pp. 364–376.

[46] A. Zhou, S. Wang, S. Wan, and L. Qi, "LMM: Latency-aware micro-service mashup in mobile edge computing environment," *Neural Computing and Applications*, Jan. 2020.

[47] E. Casalicchio and S. Iannucci, "The state-of-the-art in container technologies: Application, orchestration and security," *Concurrency and Computation: Practice and Experience*, vol. n/a, no. n/a, p. 17, Jan. 2020.

[48] "What is a Container? — App Containerization — Docker," https://www.docker.com/resources/what-container.

[49] Y. Lu, "The blockchain: State-of-the-art and research challenges," *Journal of Industrial Information Integration*, vol. 15, pp. 80–90, Sep. 2019.

[50] P. K. Sharma, M.-Y. Chen, and J. H. Park, "A Software Defined Fog Node Based Distributed Blockchain Cloud Architecture for IoT," *IEEE Access*, vol. 6, pp. 115–124, 2018.

[51] K. Xiao, Z. Gao, W. Shi, X. Qiu, Y. Yang, and L. Rui, "EdgeABC: An architecture for task offloading and resource allocation in the Internet of Things," *Future Generation Computer Systems*, vol. 107, pp. 498–508, Jun. 2020.

[52] G. S. Aujla, A. Singh, M. Singh, S. Sharma, N. Kumar, and K.-K. R. Choo, "BloCkEd: Blockchain-Based Secure Data Processing Framework in Edge Envisioned V2X Environment," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 6, pp. 5850–5863, Jun. 2020.

[53] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[54] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How Can Edge Computing Benefit From Software-Defined Networking: A Survey, Use Cases, and Future Directions," *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2359–2391, Fourthquarter 2017.

[55] C. Mouradian, F. Ebrahimnezhad, Y. Jebbar, J. K. Ahluwalia, S. N. Afrasiabi, R. H. Glitho, and A. Moghe, "An IoT Platform-as-a-Service for NFV-Based Hybrid Cloud/Fog Systems," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6102–6115, Jul. 2020.

[56] Z. Xu, W. Gong, Q. Xia, W. Liang, O. Rana, and G. Wu, "NFV-Enabled IoT Service Provisioning in Mobile Edge Clouds," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2020.

[57] T. Hong, Z. Wang, X. Luo, and W. Zhang, "State-of-the-art on research and applications of machine learning in the building life cycle," *Energy and Buildings*, vol. 212, p. 109831, Apr. 2020.

[58] P. Farhat, H. Sami, and A. Mourad, "Reinforcement R-learning model for time scheduling of on-demand fog placement," *The Journal of Supercomputing*, vol. 76, no. 1, pp. 388–410, Jan. 2020.

[59] R. Tönjes, M. I. Ali, P. Barnaghi, S. Ganea, F. Ganz, M. Haushwirth, B. Kj, and L. Vestergaard, "Real Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications," *EU FP7 CityPulse Project*, p. 5.

[60] S. M. Kaplan, "Electric Power Transmission: Background and Policy Issues," https://www.hsdl.org/?abstract&did=, Apr. 2009.

[61] W.-Y. Chiu, H. Sun, and H. V. Poor, "Demand-side energy storage system management in smart grid," in *2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm)*, Nov. 2012, pp. 73–78.

[62] R. Deng, F. Luo, G. Ranzi, Z. Zhao, and Y. Xu, "A MILP Based Two-Stage Load Scheduling Approach for Building Load's Peak-to-Average Ratio Reduction," in *2020 5th Asia Conference on Power and Electrical Engineering (ACPEE)*, Jun. 2020, pp. 771–775.

[63] S. Aman, Y. Simmhan, and V. K. Prasanna, "Energy management systems: State of the art and emerging trends," *IEEE Communications Magazine*, vol. 51, no. 1, pp. 114–119, Jan. 2013.

[64] I. Mugarza, A. Amurrio, E. Azketa, and E. Jacob, "Dynamic Software Updates to Enhance Security and Privacy in High Availability Energy Management Applications in Smart Cities," *IEEE Access*, vol. 7, pp. 42 269–42 279, 2019.

[65] I. Mugarza, J. Parra, and E. Jacob, “Cetratus: Towards a live patching supported runtime for mixed-criticality safe and secure systems,” in *2018 IEEE 13th International Symposium on Industrial Embedded Systems (SIES)*, Jun. 2018, pp. 1–8.

[66] A. H. Wiberg and D. Baer, “ZEN TOOLBOX: First concept for the ZEN Toolbox for use in the development of Zero Emission Neighbourhoods Version 1.0,” NTNU/SINTEF, Memo, 2019.

A

A Novel Edge-to-Cloud-as-a-Service (E2CaaS) Model for Building Software Services in Smart Cities

J. Robberechts, A. Sinaeepourfard, T. Goethals, B. Volckaert

presented at The First International Workshop on (3SCity-E2C) Building Software Services in Smart City through Edge-to-Cloud orchestration, in conjunction with the 21st IEEE International Conference on Mobile Data Management, Versailles, France (Online Conference)

2020 21st IEEE International Conference on Mobile Data Management (MDM)

A Novel Edge-to-Cloud-as-a-Service (E2CaaS) Model for Building Software Services in Smart Cities

Jaro Robberechts*, Amir Sinaeepourfard§, Tom Goethals‡, and Bruno Volckaert‡

Department of Information Technology*‡ - Department of Computer Science§,
Internet Technology and Data Science Lab (IDLab)‡ - imec‡,
Ghent University*‡ - Norwegian University of Science and Technology§,
Ghent, Belgium*‡ - Trondheim, Norway§
{jaro.robberechts, togoetha.goethals, bruno.volckaert}@ugent.be*‡, a.sinaee@ntnu.no§

Abstract—**The main goal of a smart city is to enhance the quality of life of its inhabitants by providing services using Information and Communications Technology (ICT) components in a city. ICT components include not only Internet of Things (IoT) data sources spread across the city, but also traditional non-IoT data sources. Managing all ICT components in a smart city can be challenging and results in many complexities. Consequently, there is a need for ICT management architectures. Traditional solutions are often based on a centralized ICT architecture using Cloud technologies. Recently, the number of ICT components, services, and their corresponding complexities are growing, leading to large-scale ICT architectures. Centralized Cloud solutions cannot cope with the ever-expanding demands of this kind of architectures. The limitations of the centralized approaches necessitate the design of a new ICT architecture, using distributed technologies, for every layer and element of the city. Many solutions for management from Edge-to-Cloud (E2C) through distributed technologies are forthcoming, including Decentralized-to-Centralized ICT (DC2C-ICT) and Distributed-to-Centralized ICT (D2C-ICT) architectures. The DC2C-ICT architecture and its components work on their own tasks and are solely communicating with a centralized platform. On the other hand, components of the D2C-ICT architecture can work together to provide the services for the citizens across different layers from E2C. Therefore, the D2C-ICT architecture is less dependent on the central Cloud-based entity, but harder to design and manage. In this paper, an "Edge-to-Cloud-as-a-Service (E2CaaS)" model is proposed together with a model on how to build efficient software services in smart cities through different layers of E2C. The most important tasks for building these services are the management of "Data/Database," "Resources," and "Network Communication and Cybersecurity issues."**

Keywords-Smart City; IoT; Edge-to-Cloud; Distributed-to-Centralized ICT architecture; Edge-to-Cloud-as-a-Service;

I. INTRODUCTION

One of the main purposes of a smart city is to offer efficient software services to its citizens, companies, organizations, and government. The most important ingredient for building such software services is data, which is widely available in every city and originates from a variety of sources. When offering a service, some data will have to be processed on a computing platform to provide value. In a traditional centralized ICT architecture, the computing platform is located in the Cloud. Recently, a novel paradigm, called Edge computing, aims to move the computing resources and storage closer to the data sources. While this concept brings many advantages like reduced latency, improved privacy, and less pressure on network communication traffic and data centers [1]. It also causes extra complexities that have to be solved efficiently.

II. RELATED WORK

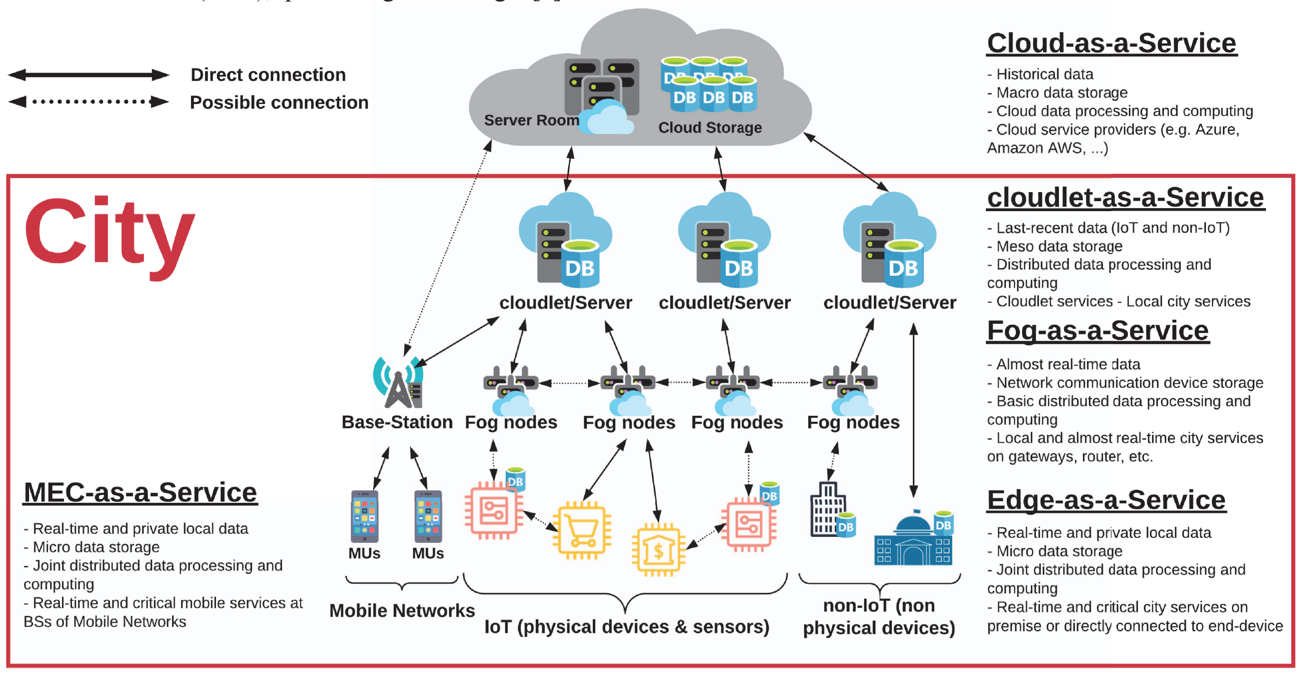
Building software services is one of the most crucial tasks in all ICT-based solutions (such as smart cities). The progression and classification of the "as-a-service (aaS)" concept have been ongoing from 1984 until the present, showing that "aaS" is a continuously evolving terminology. E.g., software is proposed as a service in 2000 and later redefined as a Cloud service such as "Infrastructure-as-a-Service (IaaS)," "Platform-as-a-Service (PaaS)," and "Software-as-a-Service (SaaS)" [2]. The classification of the Edge of smart city networks as a Service is a complex task involving different distributed to centralized technologies, such as Fog, and cloudlet. Some examples of "aaS" services through E2C orchestration are "Edge-as-a-Service (EaaS)," "Mobile Edge Computing-as-a-Service (MECaaS)," "Fog-as-a-Service (FaaS)," and "cloudlet-as-a-Service (caaS)." Fig. 1 shows the classification of the "aaS" model in a smart city through different layers of "E2C" technologies.

A. *"Cloud-as-a-Service (CaaS)"*

"CaaS" is a model that enables individuals or business companies to let specialized companies manage their Information System instead of hosting and managing the computing resources themselves [2]. Because the consumer hires the computing resources and does not have to buy and maintain expensive hardware, this model is advantageous for medium and small-sized companies who are not able to invest in this sort of equipment. The "CaaS" model is located as a top layer of the D2C-ICT architecture and is also visible as the top layer of Fig. 1. The main three models in "CaaS" are "IaaS," "PaaS," and "SaaS," as we described below.

2375-0324/20/$31.00 ©2020 IEEE
DOI 10.1109/MDM48529.2020.00079

• "IaaS" enables the provider to rent out the infrastructure used to run the software of the client, often in terms of Virtual Machines (VMs), processing or storage [3]. This gives a lot of freedom to the client because he/she can choose what kind of software stack to run on the



Direct connection
Possible connection
Server Room
DB DB DB
Cloud Storage
Cloud-as-a-Service
- Historical data
- Macro data storage
- Cloud data processing and computing
- Cloud service providers (e.g. Azure, Amazon AWS, ...)
City
DB
cloudlet/Server
cloudlet/Server
cloudlet/Server
cloudlet-as-a-Service
- Last-recent data (IoT and non-IoT)
- Meso data storage
- Distributed data processing and computing
- Cloudlet services - Local city services
Base-Station
Fog nodes
Fog nodes
Fog nodes
Fog nodes
Fog-as-a-Service
- Almost real-time data
- Network communication device storage
- Basic distributed data processing and computing
- Local and almost real-time city services on gateways, router, etc.
MEC-as-a-Service
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical mobile services at BSs of Mobile Networks
MUs
MUs
Mobile Networks
IoT (physical devices & sensors)
non-IoT (non physical devices)
Edge-as-a-Service
- Real-time and private local data
- Micro data storage
- Joint distributed data processing and computing
- Real-time and critical city services on premise or directly connected to end-device

Figure 1. Classification of "aaS" model through the D2C-ICT architecture in a smart city

infrastructure components, which include computers, storage, networking, and networking services. This type of "Cloud as a Service" is ideal when having a limited hardware budget. Some use cases for IaaS are file backups, and product design [3].

• "PaaS" is a model where a service provider rents out the infrastructure with the operating system and databases. This model is used for quickly deploying applications [3].

• "SaaS" offers applications to clients via the web. The applications can mostly be accessed through a web browser, not requiring any software installation. The client has no control over the underlying infrastructure and limited control over the application settings [3].

B. Edge of smart city networks as a Service

Edge of smart city networks as a Service can be realized by different distributed technologies [4,5] and demonstrated by different service models, "EaaS", "MECaaS", "FaaS", and "caaS". The models can be applied through three layers of city/Edge networks as shown in Fig. 1.

1) "EaaS" includes the IoT and non-IoT data sources in the Edge layer as shown in Fig. 1 and is a service model that offers services to citizens through either executing the computation tasks on the devices of the citizens themselves or through computational devices that are directly connected to the citizens' devices [6]. The services are deployed either on-premise or on a device near the citizens. If the service is not deployed on-premise, the citizens have to be able to connect via LAN to other nearby Edge devices. The Edge device can forward IoT and non-IoT data or connect directly to the higher layers (i.e. Fog/cloudlet) for further processing and storage requirements. This model allows for building "critical," "real-time," and "private" services that require fast data processing with low latency, low operation cost, and low network traffic. The data that is processed in the Edge layer, is mostly "private data" from city consumers.

2) "MECaaS" is in the left-hand side of the lowest layer of Fig. 1 and is a computing model that tries to solve the limitations of Mobile Users (MUs) by offering computing services close to the MUs. The "critical" and "almost real-time" services are deployed at the base-stations of the mobile networks [7]. Other service types can also run on the higher layers (cloudlet/Cloud).

3) "FaaS" is located at the second-lowest layer (Fog layer) of Fig. 1. Fog nodes in this layer are often network devices such as "gateways." Due to the limited computation power of the "Fog nodes", this model is mostly used for building less demanding services or data management actions like data reduction, which decreases the network traffic towards the Cloud/cloudlets, or data analysis. Plus, this study [8] proposes a FaaS technology and architecture which is built on three layers: the "Infrastructure", "Platform", and "Software" layer, identical to the available service idea for "CaaS". These three service layers only apply and execute services onto the "Fog nodes". The proposed architecture provides faster service responses and efficient use of resources.

4) "caaS" is located at the third-lowest layer (cloudlet layer) of Fig. 1 and uses cloudlet technologies. A cloudlet is a small data center/server in a box, computing device, or cluster of computers that brings the functionalities of the Cloud closer to the Edge of the network, thereby reducing the latency and network traffic. cloudlets not only include physical servers but also virtual servers. There are not many studies that specifically mention "caaS" model. However, [9] proposed a Cloud-based framework which uses cloudlets as service providers. In this framework, the main application runs on mobile devices. When a mobile device does not have the required computing power to execute a certain task, the device connects to a nearby cloudlet which will execute the task. The orchestration of the cloudlets and mobile devices is organized by a root server in the Cloud.

Focusing on the "FaaS" model, [10] defined how software services can be built at the Fog layer through a systematic literature review. The paper proposed a taxonomy that categorizes current work on Fog computing applications in smart cities. The main three categories are: "Service Objective," "Application Classification," and "Outcome Type." The "Application Classification" category seeks to place current work into groups based on their application domain (e.g. Smart Building), in a manner that the groups have minimum overlap. The "Service Objective" defines what the goals can be for a Fog application. The third and final main category of the taxonomy, the "Outcome Type" considers three main types of solutions: Architecture, Framework, and Platform.

The main limitation of the "aaS" model in smart cities is the following:

• How can we build software services in large-scale ICT networks of smart cities from smallest to largest scale? This solution must provide facilities to use different multilevel distributed and centralized technologies in combination with a different scope of ICT management strategies. Therefore, this paper presents a novel "E2CaaS" model for building software services in smart cities.

III. Software Services in Large-Scale ICT Networks of Smart Cities: ZEN as a Use Case

This section goes into detail on how to build software services in the Zero Emission Neighborhoods (ZEN). ZEN is located in Norway and researches no emission of greenhouse gas in neighborhoods through eight different city pilots [11]. Subsection III-A goes into detail about the process of building software services. Subsection III-B discusses our proposed "E2CaaS" model.

A. Different steps for building Software Services

Deciding where and how the software services should be built in a smart city can be divided into four main steps. The first step is the classification of city services. This step classifies the service and defines the service objectives. The second step is to design the output. This step goes deeper into the design of the solution. The thirds step is the implementation of the design. The final step is the efficiency measurements. The different steps can be seen in Fig. 2 and will be discussed below.

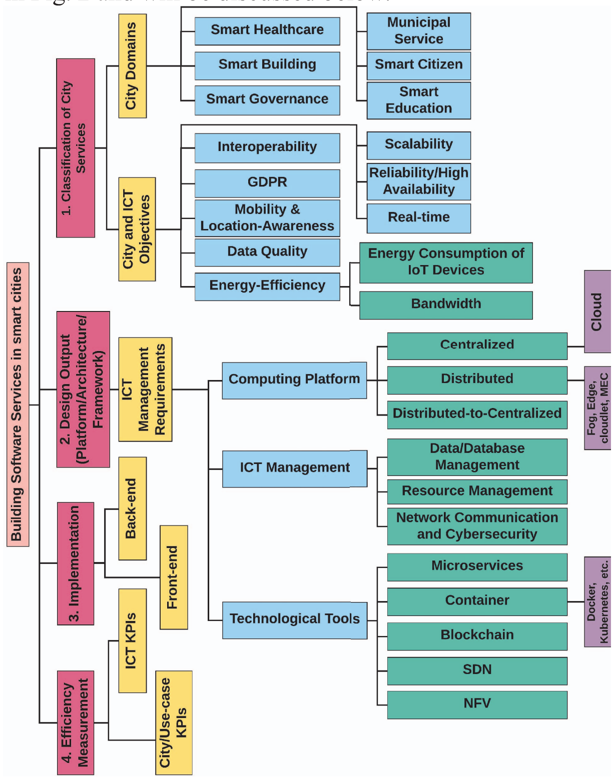


Figure 2. Building software services in smart cities

1) Classification of City Services can be organized in two steps. The first step is defining the city domains for the service, depending on the target audience and/or the type of organization. The second step is determining the city and ICT objectives for the service. These objectives mostly depend on the facilities of the domain.

We have defined some parameters, as discussed below.

• "Interoperability" means that the different computing and networking nodes from the different layers of the smart city must work together efficiently.

• "Mobility and Location-Awareness" means that services may have the possibility to handle the ability of citizen/device movement between different levels of the city, e.g., smartphone, and drones.

• "Real-time" means that the service must respond immediately, requiring a latency under a specified value. e.g., emergency health-care services.

• "Scalability" means that the services must be easily expandable when the amount of tasks increases.

• "Reliability/High Availability" means that the service should be able to withstand node failures and is often expressed in terms of uptime.

• “Data Quality” means that the services can provide certain requirements in terms of the quality of data.
• “Energy-Efficiency” has different domains in a smart city. ICT-based examples for the energy efficiency are: i) bandwidth management inside the city network communications; ii) energy consumption of IoT devices.

2) The *Design Output* will result in an Architecture, a Framework, or a Platform [10]. The main step for designing the output is exploring the ICT Management requirements. This step can be divided into three main categories: “Computing Platform,” “ICT management,” and “Technological Tools.”

a) The *“Computing Platform”* can be either “Centralized”, “Distributed” or “Distributed-to-Centralized.” Centralized computing platforms are often realised using Cloud-based technologies, meaning that all technology resources for computing, storage, and other data management phases are located in the Cloud. Distributed computing platforms involve multiple devices that are spread across the city [12]. These devices can work together to combine their computational power and perform higher demanding tasks. Distributed-to-Centralized computing platforms use both potentials of distributed and centralized technologies for processing and storage at the same time, such as Fog-to-Cloud (F2C) [12,13] and Fog-to-cloudlet-to-Cloud (F2c2C) [4,5,14].

b) “ICT Management” is an important aspect in smart cities due to the wide variety of available ICT resources and components. Currently, authors [5] mention that there are three main categories for ICT management in large-scale ICT networks of smart cities: “Data/Database management,” “Resource management,” and “Network Communication and Cybersecurity issues management.” These three concepts have to be managed efficiently and have to work together to meet the service objectives for citizens of smart cities.

• “Data/Database management” involves every activity in the life cycle of the data in a smart city, including “Data Acquisition,” “Data Preservation,” and “Data Processing.” Further details are available in [1,15].
• “Resource management” involves the efficient organization of the various types of resources in a smart city. The main challenges are related to resource discovery, resource provisioning, and resource scheduling and load balancing [6]. Resource discovery handles the identification of the resources and services inside the city and is mandatory for the system to be able to find the best-suited computing nodes for a service. This selection procedure is called resource provisioning and involves selecting the optimal computing nodes and the placement of the services and VMs onto these nodes. Finally, resource management also takes care of resource scheduling and the offloading/load balancing.
• “Network Communication and Cybersecurity issues management” is related to the communication between the ICT components. The access from and to the network resources can be controlled with tools like Network Function Virtualization (NFV) or Software-Defined Networking (SDN) [16]. These tools will be explained briefly in paragraph III-A2c. The communication between the network nodes has to be secure, as often a lot of sensitive data is processed in a smart city. Many solutions have already been proposed related to network security. E.g., Blockchain [17] or SDN [18].

c) Technological Tools: Some common tools for ICT management are listed below.

• “Microservices” are small services that are often limited to one or a couple of subtasks, which are defined by decomposing a complete task. This enables the code of the task to be run on smaller devices and to distribute the processing, making the system more responsive and fault-tolerant [19].
• “Container” technologies are lightweight virtualization technologies, enabling the deployment and execution of large-scale distributed applications on Cloud, cloudlet, Edge/Fog, and IoT platforms [20]. Containers are useful for large-scale applications because of easy life-cycle management, negligible overhead, excellent start, restart, and stop times compared to VMs and application portability. The main components of container technologies are the containers itself, the container manager, and the container orchestrator. The type of container depends on the container technology of choice, i.e. Docker, LXC, OpenVZ, etc. The container manager provides an Application Programming Interface (API) to manage the life-cycle of the containers. The orchestrator enables the application provider to manage the deployment, monitoring, and the configuration of multi-container applications. Examples of well know orchestrators are “Kubernetes” and “Docker Swarm.”
• “SDN” is an approach to networking where the data/forwarding plane and the control plane are separated by creating a virtualized control plane that manages network functions, thereby bridging the gap between service provisioning and network management. The network becomes directly programmable using Southbound Interfaces (e.g. “OpFlex”, “OpenFlow”, etc.). This type of network management results in a more flexible network that can adapt to changing network conditions, business, market and citizens' needs [16].
• “NFV” decouples Network Functions (NFs) from their specialized hardware, which enables them to be deployed on top of general-purpose hardware, greatly reducing the hardware cost for the service provider. Also, NFV enables the NFs to be easily deployed and dynamically allocated, resulting in a more scalable and flexible network [16].
• “Blockchain” originates from digital currency but is now being used in a variety of other technological domains. Due to its abilities in terms of consistency and integrity of data during transmission, its transparency, and its distributed nature, it is a viable option to protect network communication and the identity of the devices in distributed systems and IoT environments without the need for a trusted third party [17].

• “Machine Learning” can be used for the optimization of resource provisioning and improvement of the network security (e.g. anomaly detection methods).

3) “Implementation”: The thirds step is implementing the output from the previous step. This process involves selecting the right frameworks and technologies for the front-end and the back-end of the service. The result will be a service that receives data from the city or Cloud in the back-end. The back-end of an application regulates how the application works behind the scenes. The citizen can access the result of the service via the front-end, which interacts with the back-end through an API or an SDK (Software Development Kit) [21]. An API is an interface which allows different layers of an application to communicate through a collection of definitions. An SDK is a package of software development tools such as APIs, libraries, code parts, etc.

4) “Efficiency Measurements”: We consider two types of efficiency measurements in this study: ICT Key Performance Indicators (KPIs), and city/use-case KPIs. A KPI is a measurable value that indicates how well the service achieves key business objectives. KPI values are often obtained by performing simulations, measurements, or surveys about the service [14]. ICT KPIs [5] are based on the performance of “Data/Database management,” “Resource management,” and “Network Communication and Cybersecurity issues management.” ICT KPIs are factors like bandwidth and latency which are defined by the city and the domain of the application. In our use case, ZEN has defined some KPIs and assessment criteria that have to be taken into account when building software services for the ZEN and its pilots.

B. Our proposed “E2CaaS” model

There are many technological resources available in a city/Cloud environment. The orchestration of these resources must be done through network communication between the different layers. So far, city-data is produced and sent to the services that are located in the Cloud through network communication in centralized orchestrations. On the other hand, distributed technology has the facility to manage data and provide services in their own layer, close to the data-sources in a city. Current orchestrations do not focus on solutions using multilevel distributed and centralized technologies for building software services. The “E2CaaS” model can cover this gap by providing facilities to generate software services in a city concerning the “Efficiency Measurements” and “Service Objectives” of the large-scale ICT networks.

In our D2C-ICT architecture [1,4,5], we already pointed out that the cloudlet is a good option to manage the large-scale ICT networks through different KPIs and requirements. In this paper [5], we designed “I2CM-IoT (Integrated and Intelligent Control and Monitoring of IoT),” where the cloudlets act as a control layer. The motivations for positioning the control in the cloudlet are listed below.

• The cloudlets can be seen as a middleware layer between a strong Cloud and the Edge of the network;

• The cloudlets are located inside the city, close to the data sources and citizens;

• Because of the city location of the cloudlet, they are suited for applying local city policies and data privacy and GDPR;

• A cloudlet can be like an external server with more computing and storage facilities in a city and somehow in transit between Edge to Cloud;

• cloudlets can provide efficient orchestration of all physical sources management in the city and Cloud.

By moving the control of the resources to the cloudlet layer, we can provide efficient services based on the Cloud service solutions, “IaaS”, “PaaS”, and “SaaS. These service models are described below for the E2C layers, with the “I2CM-IoT” applied in the cloudlet.

• “IaaS-E2C” enables the client to rent some possibly distributed infrastructure located inside the city on which it can deploy its containers. The client has, as in the Cloud, the option to choose whatever software stack is deployed on this infrastructure. As there is a wider variety of infrastructure available in the city, the customer has more freedom to choose the type of infrastructure that suits best.

• “PaaS-E2C” is similar as in the Cloud, the service provider rents the infrastructure and controls the type of operating system and databases. This allows the client to quickly deploy applications in the city.

• “SaaS-E2C” allows service providers to rent out software services tailored to certain business use cases. The software services, located in the city, benefit from improved privacy and security and, better connectivity to citizens.

To realize these facilities when moving the control to the cloudlets, the ICT components in the city must be managed efficiently across all the E2C layers. The categories of ICT management are discussed below in a general scenario. In the specific scenario of “real-time” or “critical” services, the services should be offered directly at the Edge of the network through e.g. Edge and MEC technologies to reduce latency and improve service performance for the citizens.

• *“Data/Database management”:* There are many different databases and data types spread across the city and Cloud (e.g. IoT and non-IoT data). The cloudlet is in a position to manage all this data by connecting the Cloud and Edge database platforms. This allows for services to be built based on user requirements and capacity, e.g. “private” and “local” data is not shared outside the city if not necessary, and required data residing in clusters spread across the city and Cloud can be collected and processed efficiently.

• *“Resource management”:* A city has many heterogeneous devices offering computing resources, which should be managed efficiently. The cloudlet can orchestrate and manage these resources due to its central position between the Edge and the Cloud. By monitoring the load on the resources, the cloudlet can schedule tasks and distribute the load based on the service requirements. This idea allows for resource management based on city capacity and user requirements.

• *"Network Communication and Cybersecurity issues management":* The cloudlet is in a good position to monitor traffic in the city as well as to the Cloud. The cloudlet orchestrates the network based on measurements, activities in the city (such as cyber-attacks), and user-requirements. This concept allows for better privacy solutions and better resource allocation.

These three categories should be managed together by the cloudlet in our "E2CaaS" model to meet the "Efficiency Measurements" and "Service Objectives." The concept of an integral solution for "E2CaaS" is explained using an example scenario. A citizen requests a service which requires a private connection, processing on computing resources inside the city, and data distributed over different clusters inside the city and the Cloud. First of all, the cloudlet retrieves the necessary data from the distributed Fog, Edge, cloudlet and Cloud clusters. Due to the central location of the cloudlet layer, the cloudlets can contact the city resources, as well as the Cloud to locate and/or retrieve data. Next, the cloudlet performs the "resource management" tasks. The communication for the previous steps should be performed efficiently by monitoring traffic over the city networks while respecting the privacy and the security of the citizen their data. The orchestration decisions made in the cloudlet are based on a multi-attributed cost model, composed of ICT and city/use case KPIs. In the case of the ZEN center, an integral solution must be directed towards particular services for their pilots and the specific requirements of those pilots.

CONCLUSION

In this paper, we proposed the "E2CaaS" model for building software services in large-scale ICT networks of smart cities. We envisage that the usage of different multilevel distributed and centralized technologies in combination with a different scope of ICT management strategies may allow us to obtain additional efficiency increment when creating efficient software cities in smart cities. These ICT technology management strategies include management of "Data/Database," "Resource," and "Network communication and Cybersecurity issues."

As part of our future studies, we will discover more options related to developing our "E2CaaS" by focusing on the ZEN center and its pilots' requirements. Our interests are to design, implement, and operate an integral solution for building Software Services in smart cities through "E2C" orchestration.

ACKNOWLEDGEMENT

This paper has been written within the Zero Emission Neighborhoods in smart cities (FME ZEN). The authors gratefully acknowledge the support from the ZEN partners and the Research Council of Norway.

REFERENCES

[1] A. Sinaeepourfard, et al., "D2C-DM: Distributed-to-Centralized data management for smart cities based on two ongoing case studies," Intelligent Systems and Applications, IntelliSys, Advances in Intelligent Systems and Computing, vol 1038, 2019.

[2] I. Adamu, et al., "Cloud computing as a service era: origins, current and future trends," IOSR Journal of Computer Engineering (IOSR-JCE), vol. 21, pp. 34-39, 2019.

[3] A. G. Prajapati, et al., "All About Cloud: A Systematic Survey," International Conference on Smart City and Emerging Technology (ICSCET), India, pp. 1-6, 2018.

[4] A. Sinaeepourfard, et al., "D2C-SM: Designing a Distributed-to-Centralized software management architecture for smart cities," Digital Transformation for a Sustainable Society in the 21st Century, Lecture Notes in Computer Science, vol 11701, 2019.

[5] A. Sinaeepourfard, et al., "A Distributed-to-Centralized smart technology management (D2C-STM) model for smart cities: a use case in the zero emission neighborhoods," IEEE International Smart Cities Conference (ISC2), Morocco, pp. 760-765, 2019.

[6] A. Yousefpour, et al., "All one needs to know about fog computing and related edge computing paradigms: a complete survey," Journal of Systems Architecture, vol. 98, pp. 289--330, 2019.

[7] I. Elgendy, et al., "Resource allocation and computation offloading with data security for mobile edge computing," Future Generation Computer Systems, 2019.

[8] N. Chen, et al., ``Fog as a service technology,'' IEEE Communications Magazine, vol. 56, no. 11, pp. 95-101, 2018.

[9] R. Kumar, S.K. Yadav, "Scalable key parameter yield of resources model for performance enhancement in mobile cloud computing." Wireless Pers Commun 95, 3969–4000, 2017.

[10] G. Javadzadeh, G. Rahmani, "Fog computing applications in smart cities: a systematic survey," Wireless Networks, vol.26, pp. 1433-1457, 2020.

[11] Available: https://fmezen.no/about-us/.

[12] A. Sinaeepourfard, et al., "Data Preservation through Fog-to-Cloud (F2C) data management in smart cities," IEEE 2nd International Conference on Fog and Edge Computing (ICFEC), pp. 1--9, 2018.

[13] A. Sinaeepourfard, et al., "Fog-to-Cloud (F2C) data management for smart cities," Saiconference on FTC, Canada, 2017.

[14] A. Sinaeepourfard, et al., "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management architecture in smart city," IEEE 5th World Forum on Internet of Things (WF-IoT), Ireland, pp. 590-595, 2019.

[15] A.Sinaeepourfard, et al., "A data lifecycle model for smart cities," IEEE conference on ICTC, South of Korea, 2016.

[16] A. A. Barakabitze, et al., "5G network slicing using SDN and NFV: a survey of taxonomy, architectures and future challenges," Computer Networks, vol. 167, 2020.

[17] P. K. Sharma, et al., "A software defined fog node based distributed blockchain cloud architecture for IoT," IEEE Access, vol. 6, pp. 115-124, 2018.

[18] A.Sinaeepourfard, et al., "Cybersecurity in large-scale smart cities: novel proposals for anomaly detection from edge to cloud," The International Conference on Internet of Things, Embedded Systems and Communications, Tunisia, 2019.

[19] L. Wu, et al., "MicroRCA: root cause localization of performance issues in mircoservices," IEEE/IFIP Network Operations and Management Symposium, Hungary, Apr. 2020.

[20] E. Casalicchio, S. Iannucci, "The state-of-the-art in container technologies: application, orchestration and security," Concurrency and Computation: Practice and Experience, 2020.

[21] R. Tönjes, et al., "Real Time IoT Stream Processing and Large-scale Data Analytics for Smart City Applications," EU FP7 CityPulse Project.