Research Centre on
ZERO EMISSION
NEIGHBOURHOODS
IN SMART CITIES

ZEN DATA MANAGEMENT AND MONITORING

REQUIREMENTS AND ARCHITECTURE

ZEN REPORT No. 34 – 2021

Amir Sinaeepourfard, John Krogstie, Sobah Abbas Petersen | NTNU

Research Centre on
ZERO EMISSION
NEIGHBOURHOODS
IN SMART CITIES

ZEN Report No. 34

Amir Sinaeepourfard (NTNU), John Krogstie (NTNU), Sobah Abbas Petersen (NTNU)

ZEN Data management and monitoring. Requirements and architechture

Keywords: Large-scale ICT architecture, Distributed to Centralised Data Management, KPI-based monitoring system
Emneord: Storskala IKT-arkitektur, distribuert til sentralisert datahåndtering, KPI-basert monitoreringssystem
ISBN 978-82-536-1719-0
Norwegian University of Science and Technology (NTNU) | www.ntnu.no
SINTEF Community | www.sintef.no

https://fmezen.no

Preface

Acknowledgements

This report has been written within the Research Centre on Zero Emission Neighbourhoods in Smart Cities (FME ZEN). The author gratefully acknowledge the support from the Research Council of Norway, the Norwegian University of Science and Technology (NTNU), SINTEF, the municipalities of Oslo, Bergen, Trondheim, Bodø, Bærum, Elverum and Steinkjer, Trøndelag county, Norwegian Directorate for Public Construction and Property Management, Norwegian Water Resources and Energy Directorate, Norwegian Building Authority, ByBo, Elverum Tomteselskap, TOBB, Snøhetta, Asplan Viak, Multiconsult, Sweco, Civitas, FutureBuilt, Hunton, Moelven, Norcem, Skanska, GK, Nord-Trøndelag Elektrisitetsverk - Energi, Smart Grid Services Cluster, Statkraft Varme, Energy Norway, Norsk Fjernvarme and AFRY.

The Research Centre on Zero Emission Neighbourhoods (ZEN) in Smart Cities

The ZEN Research Centre develops solutions for future buildings and neighbourhoods with no greenhouse gas emissions and thereby contributes to a low carbon society.

Researchers, municipalities, industry and governmental organizations work together in the ZEN Research Centre in order to plan, develop and run neighbourhoods with zero greenhouse gas emissions. The ZEN Centre has nine pilot projects spread over all of Norway that encompass an area of more than 1 million m^2 and more than 30 000 inhabitants in total.

In order to achieve its high ambitions, the Centre will, together with its partners:

- Develop neighbourhood design and planning instruments while integrating science-based knowledge on greenhouse gas emissions;
- Create new business models, roles, and services that address the lack of flexibility towards markets and catalyze the development of innovations for a broader public use; This includes studies of political instruments and market design;
- Create cost effective and resource and energy efficient buildings by developing low carbon technologies and construction systems based on lifecycle design strategies;
- Develop technologies and solutions for the design and operation of energy flexible neighbourhoods;
- Develop a decision-support tool for optimizing local energy systems and their interaction with the larger system;
- Create and manage a series of neighbourhood-scale living labs, which will act as innovation hubs and a testing ground for the solutions developed in the ZEN Research Centre. The pilot projects are Furuset in Oslo, Fornebu in Bærum, Sluppen and Campus NTNU in Trondheim, an NRK-site in Steinkjer, Ydalir in Elverum, Campus Evenstad, NyBy Bodø, and Zero Village Bergen.

The ZEN Research Centre will last eight years (2017-2024), and the budget is approximately NOK 380 million, funded by the Research Council of Norway, the research partners NTNU and SINTEF, and the user partners from the private and public sector. The Norwegian University of Science and Technology (NTNU) is the host and leads the Centre together with SINTEF.

https://fmezen.no
@ZENcentre
FME ZEN (page)

Sammendrag

Behandling av data i nullutslippsområder

For å understøtte og følge opp nullutslippsområder og smarte bærekraftige byer trenger vi nye løsninger samt oppfølging av løsningene for å avgjøre om man når blant annet utslippsmål mens man understøtter et godt bymiljø. I mange tilfeller betyr det innsamling, forvaltning og distribusjon av data og informasjon. Tradisjonelt har man tenkt seg at dette best kan gjøres ved å samle inn data i en skyløsning som alle kan ha tilgang til, men det er ofte ikke den mest hensiktsmessige måten å gjøre dette på.

Denne rapporten beskriver hensyn og krav til en IKT-arkitektur og dens databehandlingsfunksjoner i en ZEN- og smartbykontekst. Det er en leveranse fra Task 1.1.2 Information Management of Big Data for å oppnå at ZEN, som er en del av WP1, Analytical Framework for design og planlegging av ZEN.

Dette forslaget oppfyller seks mål:

1. For det første beskriver den hvordan IKT-arkitekturer kan utformes i smarte byer.
2. For det andre beskriver den hvordan en datahåndteringsarkitektur kan tilpasses ulike IKT- arkitekturer i en smart by.
3. For det tredje beskriver den hvordan man integrerer store Internet of Things (IoT)-nettverk i IKT-arkitekturen til byens overordnede IKT-styring
4. For det fjerde beskriver den hvordan et monitoreringssystem kan integreres i IKT-arkitekturen.
5. For det femte beskriver den hvordan oppfølging av KPIene (Key Performance Indicators) identifisert i ZEN-senteret kan støttes av den foreslåtte IKT- og databehandlingsarkitekturen.
6. Til slutt beskriver den hva som er potensielle ekstra fordeler med den foreslåtte IKT-arkitekturen, for eksempel utvikling av programvaretjenester og forbedring av bruken av maskinlæringsteknikker (ML) og kunstig intelligens (AI) i smarte byer.

Funnene i dette forslaget som er basert på en rekke vitenskapelige artikler kan være gunstige for fremtidige smarte byer i forbindelse med:

• Design av storskala IKT-arkitektur inkludert IoT-nettverk i en smart by.
• Organisere og administrere innhentede bydata fra forskjellige kilder, med fokus på IoT-kilder.
• Designe et KPI-basert monitoreringssystem i smartbyen ved hjelp av store IoT-nettverk.

Summary

This report details considerations and requirements for an ICT architecture and its data management features within a ZEN and smart cities context.
It is a deliverable from Task 1.1.2 Information Management of Big Data to Achieve ZEN being part of WP1, Analytical Framework for Design and Planning of ZEN.

This proposal fulfills six goals:

- First, it describes how Information and communications technology (ICT) architectures can be designed in smart cities.
- Second, it describes how a data management architecture can be fitted to different ICT architectures in a smart city.
- Third, it describes how to integrate smart cities' large-scale Internet of Things (IoT) networks in the ICT architecture of the city large-scale management of ICT and their data management can work in smart cities' large-scale Internet of Things (IoT) networks.
- Fourth, it describes how a monitoring system can be integrated in the ICT architecture.
- Fifth, it describes how following up the Key Performance Indicators (KPIs) identified in the ZEN center can be supported by the proposed ICT and data management architecture.
- Finally, it describes what are potential extra benefits of the proposed ICT architecture, e.g., developing software services and improving the use of machine learning (ML) and Artificial Intelligence (AI) techniques in smart cities.

The findings of this proposal can be beneficial for future smart cities in connection to:

- Designing large-scale ICT architecture including IoT networks in a smart city.
- Organize and manage obtained city-data from different sources, with a focus on IoT-sources.
- Designing a KPI-based a monitoring system in smart city utilizing large-scale IoT networks.

Contents

Preface 3

Sammendrag 4

Summary 5

List of Tables: 9

List of Abbreviations: 10

1 Introduction 11

1.1 Research questions 12

1.2 Project objectives 12

2 ICT Architecture 14

2.1 Smart city ICT architecture 14

2.1.1 ICT reference architecture for smart cities 14

2.1.2 ICT reference architecture in a smart city 15

2.2 Smart city ICT architecture: Focus on Technical View (Vertical Landscape) and Management Blocks (Horizontal Landscape) 17

2.3 Our proposed ICT architecture in smart cities 19

2.3.1 DC2C-ICT Architecture 19

2.3.2 D2C-ICT Architecture 20

2.3.3 Hybrid Architecture 24

3 Data Management 25

3.1 Data management in Big Data Management context 26

3.2 Data management architectures in smart cities 26

3.2.1 The CDM architecture 26

3.2.2 The DC2C-DM Architecture 27

3.2.3 The D2C-DM Architecture 29

3.2.4 The F2c2C-DM Architecture 30

4 A Proposed ICT Architecture and its Data Management Architecture for ZEN Centre 32

4.1 ZEN Background 32

4.2 Conceptual Levels of ZEN Study 32

4.3 ZEN Data Types 33

4.4 Interconnection of ZEN Definition, ZEN KPIs and ZEN Toolbox 35

4.5 Large-Scale ICT management for ZEN center 35

4.5.1 Data Management Architecture 36

4.5.2 Software Services architecture 37

4.5.3 Monitoring system and their related ICT controller 39

5 Future Works and Efficiency of our Proposed D2C-ICT Architecture 41

5.1 Developing Software Services for the ZEN 41

5.2 Applying multilevel ML and AI Techniques within and across F2c2C-ICT architecture 42
6 Implementation 44
6.1 Overview of the ZEN pilot's implementation 44
6.1.1 Use case 1: Micro Level (The Living Lab) 44
6.1.2 Use case 2: Meso Level (Campus Evenstad) 44
6.1.3 Use case 3: Macro Level (partners and/or smart city users) 45
6.2 Scenario description for our ICT architecture proposal 46
6.2.1 Large-Scale ICT Management Architecture 48
6.2.2 User Interface and results 52
6.2.3 Results and discussions 55
Discussions 65
7 Conclusion 67
Acknowledgment 67
References 68
A Appendix 70
A.1 Contribution of this proposal with ZEN centre up to December 2020 70
A.2 Recent publications 70
A.3 Recent activity and dissemination of the results 71
A.4 Recent Supervision/Co-Supervision Activities 72

List of figures

Figure 1 Conceptual levels addressed in this report 12
Figure 2 Various views on the ICT Reference architecture for smart cities [7] 15
Figure 3 Proposal for the ICT Reference Architecture for Smart Cities [7] 16
Figure 4 Proposal for the ICT Reference Architecture for Smart Cities [13] 18
Figure 5 Our proposed DC2C-ICT architecture for smart cities........ 20
Figure 6 Our proposed D2C-ICT architecture in the smart city 21
Figure 7 Our proposed hybrid ICT architecture in the smart city based on data-source formats 25
Figure 8 Our proposed hybrid architecture in the smart city based on city districts 25
Figure 9 CDM architecture through centralized ICT architecture in smart cities [9] 27
Figure 10 Data management strategies and steps in the Cloud platform [26] 27
Figure 11 The DLC model [25, 36]........ 28
Figure 12 The F2C-DM architecture in smart cities [12]........ 29
Figure 13 The c2C-DM architecture for Smart Cities 30
Figure 14 The F2c2C-DM Architecture for Smart Cities........ 31
Figure 15 Conceptual levels around a city and their neighborhoods and buildings 33
Figure 16 Data types in ZEN center through the ZEN conceptual models 34
Figure 17 Different data types in smart cities and ZEN center 35
Figure 18 Interconnection of ZEN definition, KPIs and toolbox [40] 35
Figure 19 The F2c2C-DM architecture through the smart city scenario [11] 37
Figure 20 The ZEN pilots city location 37
Fig. 21. Building software service through F2c2C-ICT architecture for ZEN [41]........ 38
Fig. 22. The proposed monitoring system and related I2CM-IoT Management Block [42] 40
Figure 23 Four main steps for building software services in smart cities........ 41
Figure 24 F2c2C-DM architecture through the living lab scenario........ 44
Figure 25 F2c2C-DM architecture through the Campus Evenstad scenario........ 45
Figure 26 F2c2C-DM architecture through the smart city scenario 46
Figure 27 Our scenario description in two different cities in Norway 47
Figure 28 Our scenario for two apartments in a building network 47
Figure 29 Proposed scenario for data/database management........ 48
Figure 30 Our scenario for data discovery 49
Figure 31 Internal architecture of an I2CM-IoT 49
Figure 32 Proposed algorithm for the cost model ranking calculation 50
Figure 33 New forwarding rule added to the Oslo ICN relay node 51
Figure 34 “Configuration” page for the cost model implementation........ 53
Figure 35 “Search” Page for the cost model Implementation 53
Figure 36 “Data Collection” page for the ZEN network discovery........ 54
Figure 37 “Network Monitoring” page for the ZEN network discovery 55
Figure 38 Average response time for one real-time data packet........ 56
Figure 39 Latency for one real-time data packet from three different levels........ 57
Figure 40 Response times for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule........ 59
Figure 41 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the micro level........ 59
Figure 42 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the meso level........ 60
Figure 43 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the macro level........ 61
Figure 44 Response time for one historical data packet 62
Figure 45 Latency for one historical data packet from three different levels........ 63
Figure 46 Response time for different packet sizes 64

List of Tables:

Table 1 Overview of the seven ZEN pilot projects [1] 32
Table 2 FIB for the cloudlet node in Oslo..51
Table 3 Latency in milliseconds for requests for one real-time data packet to cloud and cloudlet layer controllers...56
Table 4 Latency in milliseconds for requests for one last-recent data packet to cloud and cloudlet layer controllers...61
Table 5 Latency in milliseconds for requests for one historical data packet to cloud and cloudlet layer controllers...63

List of Abbreviations:

Number	Abbreviation	Description
1	c2C-DM	cloudlet-to-Cloud Data Management
2	CDM	Cloud Data Management
3	COSA-DLC	Comprehensive Scenario Agnostic Data LifeCycle
4	D2C-DM	Distributed-to-Centralized Data Management
5	DLC	Data LifeCycle
6	F2C	Fog-to-Cloud
7	F2C-DM	Fog-to-Cloud Data Management
8	F2c2C-DM	Fog-to-cloudlet-to-Cloud Data Management
9	IoT	Internet of Things
10	SCC-DLC	Smart City Comprehensive Data LifeCycle
11	ZEN	Zero Emission Neighborhoods
12	ICT	Information and communications technology
13	KPIs	Key Performance Indicators
14	ML	Machine Learning
15	AI	Artificial Intelligence
16	GDPR	General Data Protection Regulation
17	DC2C-ICT architecture	Decentralized-to-Centralized ICT architecture
18	DC2C-DM architecture	Decentralized-to-Centralized Data Management Architecture
19	D2C-ICT architecture	Distributed-to-Centralized ICT architecture
20	D2C-DM architecture	Distributed-to-Centralized Data Management Architecture
21	CDM-DM architecture	Centralized Data Management architecture
22	F2c2C-ICT architecture	Fog-to-cloudlet-to-Cloud ICT architecture
23	F2C-ICT architecture	Fog-to-Cloud ICT architecture
24	D2C-SM	Distribtued-to-Centralized software management
25	DLC	Data LifeCycle
26	SCC-DLC	Smart City Comprehensive Data LifeCycle
27	I2CM-IoT	Integrated and Intelligent Control and Monitoring of IoT
28	MAUT	Multi-Attribute Utility Theory
29	SDN	Software-Defined Networking
30	ICN	Information-Centric Networking
31	GHG	GreenHouse Gas
32	CDN	Content Delivery Network

1 Introduction

Smart cities are an excellent example of a situation where input from ICT and software developers, citizens and business requirements, and city manager policies need to be combined. Those contributions provide several opportunities to discuss developing software services in smart cities through novel ICT technologies and their related ICT architectures. This ICT architecture may address design by understanding different smart cities' future elements, such as city-resource management, IoT, AI, cybersecurity, and Edge-to-Cloud computing orchestration. This proposal also highlights other challenges in developing efficient software services in a city by designing ICT architecture and platform, such as its data management, resource management, network management, and cybersecurity issues.
Edge computing technologies provide the development of applications in smart cities that can run independently of an Internet connection. Networking a number of these boxes together builds a distributed computing resource spread across parts of a city. Various edge servers can span the area and provide a mesh to form a network. Based on an open-source software platform, the solution allows applications and software services typically run centrally to run locally at the edge of networks. The result is to provide several facilities for developing an ICT architecture, as shown below:

- Autonomous operation if backhaul coverage fails.
- Lower latency because applications are closer to the data source of information.
- Reduced load on backhaul links as more data is being processed locally and distributed.
- An open service platform is allowing third-party developers to build innovative smart city applications.
- ICT technologies present an attractive and resilient platform for cities, while at the same time decreasing backhaul costs and enhancing service performance.
- A third-party intercepting confidential business data on the public internet which introduce a business at risk can also be tackled through Edge Computing.

The overall goal of the Research Centre on Zero Emission Neighborhoods in Smart Cities (FME-ZEN Centre) is to enable the transition to a low carbon society by developing sustainable neighborhoods with zero greenhouse gas (GHG) emissions. It will speed up decarbonization of the building stock (existing and new), use more renewable energy sources, and create positive synergies between the building stock, energy, ICT, mobility systems, and citizens [1].
It includes how sustainable neighborhoods are to be designed, built, transformed, and managed for the GHG emission reduction.
A range of ICT-based technologies will support the ZEN goals and the implementation work in the pilots as well as the monitoring of the overall project progress. In ZEN Pilot Projects and Living Labs, large amounts of IoT- and non-IoT data will be produced, collected, aggregated, and distributed to support the goals of achieving a zero-emission neighborhood in a smart city. Politicians, planners, developers, communities, and citizens will use the data to support informed decisions towards reducing emissions.
This report examines the requirements coming from this specific task, as shown in Fig. 1. The rest of the report is structured based on Fig. 1 content and their tasks, which are “logical solution for ICT architecture” mainly addressed in Section 2, “Data management architecture as part of the proposed ICT architecture” mainly focus on Section 3 and 4, and “implementation ” mainly addressed in Section 5 and 6, where we also highlight future work

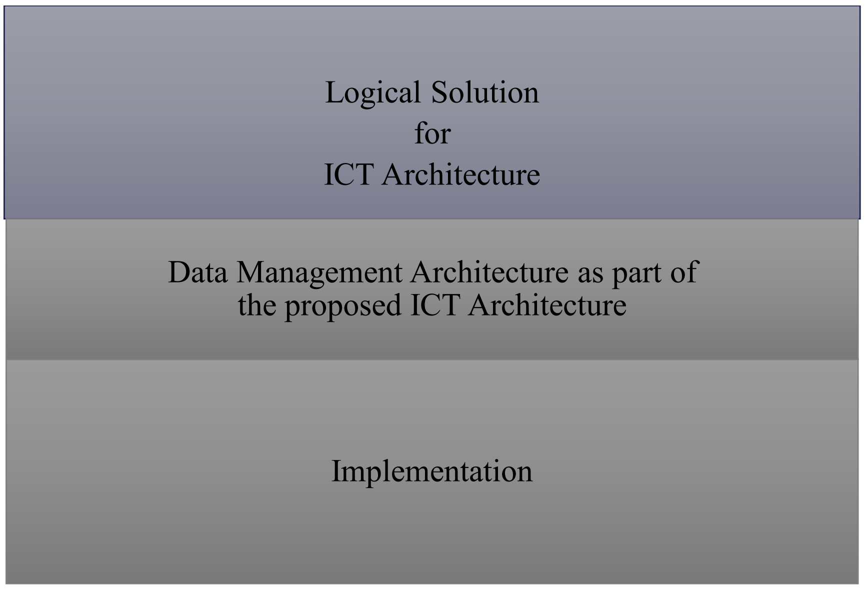

Logical Solution
for
ICT Architecture
Data Management Architecture as part of the proposed ICT Architecture
Implementation

Figure 1 Conceptual levels addressed in this report

1.1 Research questions

We have observed the below challenges in the accessible ICT architectures in smart cities:

- Designing large-scale ICT architecture: As a considerable number of data and services are created and taken into use in a smart city, there is a necessity to organize all technology resources, obtained city-data, software services, network communications, and cybersecurity concerns at all scale of the city;
- Proposing a unified and extensive ICT architecture: Most of the recommended ICT architectures in smart cities have been designed to organize individual aspects in a city; therefore, there are no suggested architectures for smart cities than include all management of large-scale smart city networks in a unified ICT architecture. Some examples of management levels of ICT technology are resources, data, network communication, and software services;
- Which ICT architecture may you use as a basis for your smart city? Using 21st century technologies, there is an opportunity to create multiple ICT architectures in your smart city by utilizing diverse ICT architecture proposals. E.g., Centralized or decentralized-to-centralized ICT (DC2C-ICT) or distributed-to-centralized ICT (D2C-ICT) architecture. Though, there is always an important question on "which one is the most suitable design option for your smart city?"
- With a focus on creating DC2C-ICT and D2C-ICT architecture in smart cities, numerous calls for new research on improving the recent ICT architectures management in a smart city are available including new approaches to data management.

1.2 Project objectives

This proposal's overarching goal is to design a "data management architecture utilizing ICT Networks of Smart City from neighbourhoods to city-scale based on edge-to-cloud orchestration" taking into account the Zero Emission Neighbourhoods (ZEN) requirements and it's KPIs.
Two main objectives are defined.

- Objective 1: Design an effective large-scale ICT management architecture including a data management architecture from neighbourhoods to the full scale of a city in a smart city through edge-to-cloud orchestration.
  This architecture may provide an integral architecture solution for developing software services in smart cities fitted with ZEN requirements.
  - To achieve this objective, it is necessary to find an integral architecture solution for the management of data/databases and data privacy issues, resources, and network communication and related cybersecurity issues in smart cities.

  - Proposing secured software services through real-time and fast data processing in ICT networks of smart cities.

- Objective 2: Discuss the facility of our proposed ICT architecture for future smart cities' demands, such as developing software services, and improving the use of ML and AI techniques in large-scale ICT networks of smart cities.

2 ICT Architecture

Architecture [2] is “a definition of the structure, relationships, views, assumptions and rationale of a system.” The ICT field is over-using the architecture term. The architecture may include the technical management of an ICT solution for different companies or business scenarios ranging from the structure of information to technology delivery [3, 4]. The core objectives of ICT architecture and systems may suggest providing the requirements and efficiencies to technology distribution. The main tangible building blocks of ICT architecture are hardware, software, network devices, and communication appliances. Mechanisms for organizing these blocks according to pre-defined procedures with set parameters guarantee your ICT systems' friction-free and efficient operation. Configuration and capacity planning, change management, standardization of all components as well as policies and procedures, backup of data, service level agreements with agreed availability and reliability parameters, disaster recovery planning, contingency planning, risk management, etc. are some examples of these parameters and their related mechanisms [5]. Finally, ICT architecture and systems objectives may suggest providing the requirements and efficiencies.

This section is divided into two main subsections. First, we discuss smart city ICT architecture. Second, we focus on the technical view as part of the vertical landscape of smart city ICT architecture in smart cities and management blocks as part of the horizontal landscape of smart city ICT architecture in smart cities. Finally, we explain our proposed ICT architecture for smart cities based on edge-to-cloud orchestration.

2.1 Smart city ICT architecture

Designing ICT architecture in smart cities is trying to achieve some main goals [6]. One of the critical goals is to offer an ICT architecture for the smart city that highlight the new ICT equipment to improve smart cities' future performance and quality of life of citizens and make urban development more sustainable.

This subsection is divided into two main subsubsections, including ICT reference architecture for smart cities and ICT reference architecture landscape for smart cities.

2.1.1 ICT reference architecture for smart cities

Numerous views [7] on the emerging smart city ICT reference model are considered. In [7], as illustrated in Fig. 2, the authors suggested three main blocks and their related layers for their smart city ICT reference model. Those main blocks are "organizational," "informational," and "technical" views, as described below briefly [7].

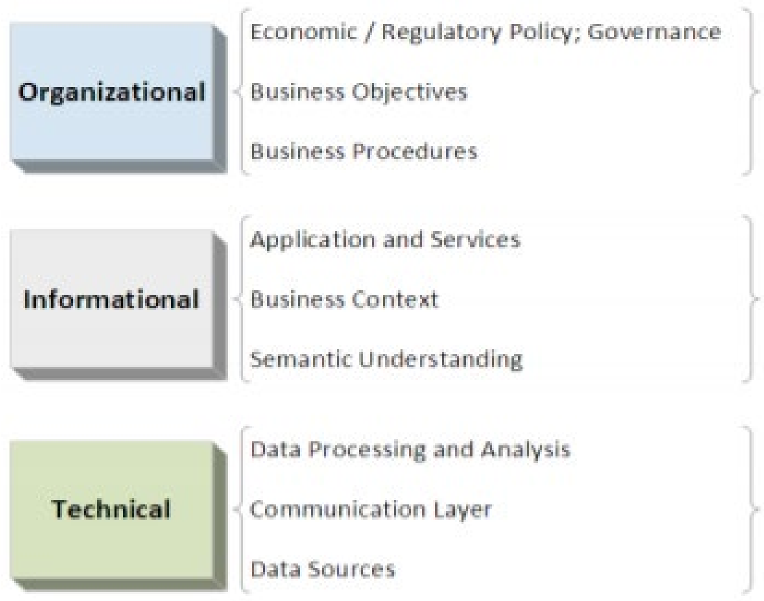


Figure 2 Various views on the ICT Reference architecture for smart cities [7]

Technical View Block

The technical view block includes the raw data sources and the communication and computational means to fuse data and then provide it for data processing and analysis. This data may be either provided commercially or being accessible over Open Data portals/platforms. Through Data Processing and Analysis, various aspects of the heterogeneous, raw data are connected and improved utilizing complex processes to convert information, whereby pushing it into the Informational view scope.

Informational View Block

The informational view block provides the refinement, structuring, and extra enrichment of the information including semantic relations and understanding of the raw data and producing information items. Thereby, various data/information pieces can be put together to understand the possible influences and implications in complicated situations deeply. This additional processing can be provided as a service. Moreover, the semantically enriched data/information is related to a business concept that enables high-level applications and software services for smart cities.

Organizational View Block

The organizational view block aims to coordinate all related issues to technical and informational aspects based on business models and their business procedures and objectives as well as different governance and regulations' perspectives.

2.1.2 ICT reference architecture in a smart city

This sub section is separated into two main subsections. Based on Fig. 3 [7], we describe two different landscape of an ICT reference architecture.

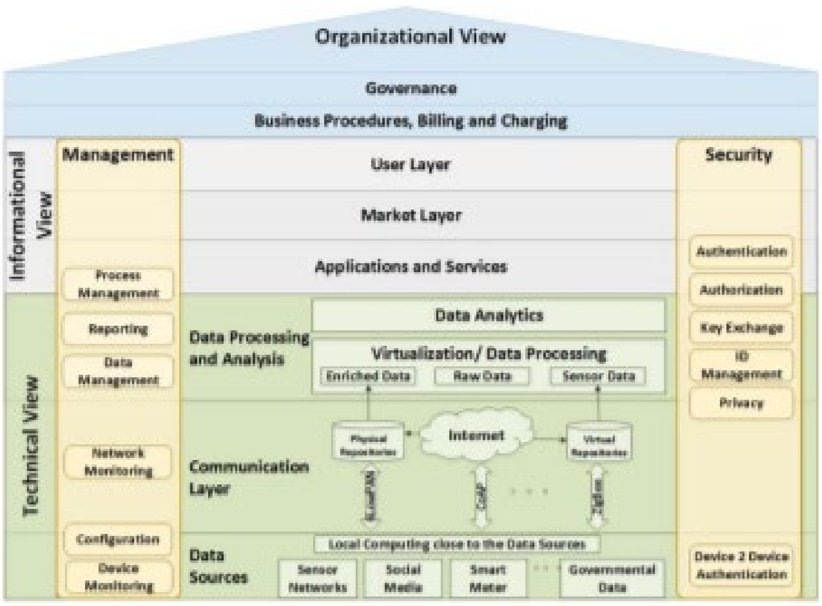


Figure 3 Proposal for the ICT Reference Architecture for Smart Cities [7]

Horizontal Landscape

Horizontal landscape of ICT reference architecture for smart cities suggested different blocks, including technical, informational, and organizational views, as described below:

- ***Technical View block***

The technical view block is in the bottom layer of Fig. 3. This layer consists of "data source," "communication," and "data processing and analysis" layers.

The data sources Layer includes the multiple city-data sources capturing and accumulating city-data within a city.

The communication layer involves all the sorts of network communication and infrastructure expected to collect/gather the data from the Data Sources Layer, send it to data repositories, and prepare for further processing.

The data processing and analysis layer may receive all city-data within the Data Sources Layer and be reached through the Communication Layer to be processed to achieve the required information. Data from various distributed city-data sources is accessible from a single point of access and processing in this kind of ICT Reference Architecture design [14].

- ***Informational View block***

The informational view block is in the mid-layer of Fig. 3. This layer has different layers, "applications and services," "market," and "user" layers.

The applications and services layer will collect and deliver the required information from the layer below to be used in various smart city applications and services. This layer is also responsible for the data processing and analysis layer.

The market layer is involved with marketplaces to support citizens and organizations in discovering, purchasing, and deploying public applications and services in different city domains using online platforms.

The user layer involves several downloading apps for related devices and their usage by the citizens. The devices comprise smartphones, personal computers, and other technical appliances. The citizens might be using the apps on those devices or install specific required services through the related marketplaces. This layer provides facilities for the ICT Reference architecture concerning how the (open) data, converted into information, is used within a smart city.

- ***Organizational View block***

The organizational view block is in the topmost layer of Fig. 3. This block includes the "governance" and "business procedures, billing, and charging" layers.
Governance describes the interaction of processes, information, rules, structures, and norms [1, 8]. This interaction leads behavior towards stated purposes to help governing activities. This interaction may also impact people's collections and governance infrastructure through managing technologies, people, policies, practices, resources, social norms, and information. A smart city requires a smart governance infrastructure, making together various stakeholders, worked out processes, rules and policies, and helping tools in accelerating growth and adaptability of smart services within the city.

Vertical Landscape

Based on the vertical landscape, across all blocks and layers of the ICT reference architecture in Fig. 3 is "management" and "security" blocks.
The management block must be depicted across various domains and all blocks and their layers in Fig. 3. Management includes numerous features, including "devices monitoring," "configuration," "network monitoring," "data management," "reporting," and "process management."
Management issues is a crucial task for all layers of the blocks in Fig. 3. For instance, the multiple heterogeneous devices and networks' proper configuration must be considered on the lower layers of data sources and communication in Fig. 3. Some of this configuration are characteristics such as hostnames, routing, quality of service, etc.
Security is a significant and complicated issue in smart cities. Security goes across multiple domains and different aspects, such as “authentication,” “privacy,” “ID management,” “key exchange,” and “authorization.” All the blocks and their related layers in Fig. 3 must repeatedly consider security and their related issues.

2.2 Smart city ICT architecture: Focus on Technical View (Vertical Landscape) and Management Blocks (Horizontal Landscape)

This report focuses on the technical view being one of the main parts of the vertical landscape and management block as one of the main blocks of the horizontal landscape in Fig. 3. We also note that the base of Fig. 3 is relevant for a cloud-based solution. We realize that the new generation of ICT reference architecture is based on edge-to-cloud orchestration and new city requirements as data privacy, General Data Protection Regulation (GDPR), etc.
Three main ICT architecture proposals are designed for smart cities, including centralized, decentralized-to-centralized, and distributed-to-centralized. Further aspects of these three ICT architectures in smart cities are as follows:
The smart city centralized ICT architecture may receive, store, and access all obtained city-data in a centralized platform. The city-data sources can create city-data varieties with different physical and non-physical data sources to send those city-data to the centralized platform to support further citizen and business requirements. Cloud-based platforms and solutions are often proposed for a centralized platform [9, 10].
The cloud-based solutions are useful for different management blocks of the horizontal landscape in a centralized platform in a smart city, as shown in the left-hand side of Fig. 4. So, the city-data collection has occurred inside the city within and across IoT and non-IoT data sources. In most cases, cloud technologies may be located outside of the city or country or continent, to be responsible for all management tasks, including data management, resource management, and software/service management. Also, network management and its cybersecurity issues have been organized from city to cloud-based technologies.

Numerous benefits are driven into attention by applying cloud solutions. Tremendous cost-saving potential for the businesses and city administrators and distributing excellent technology resources to the citizens are examples of those benefits [9, 10]. Despite this, some of the drawbacks of Cloud technologies are data privacy, cybersecurity concerns, and geographical access to Cloud technologies—the main drawback of cloud technologies mainly related to the city-data generated by IoT devices. The generated city-data was relayed back to a central network server, ordinarily housed in a Cloud technologies data center. While data is processed, data returned to the devices on the network's edge with more extra delay and network movement [9, 10].

Distributed technologies provide facilities to extend all vertical and horizontal landscapes with their related layers and blocks of smart city ICT reference architecture from edge to cloud orchestration. Various distributed technologies may apply for the local and centralized processing and storage orchestration at the same time through decentralized-to-centralized and distributed-to-centralized ICT architectures in smart cities. Fog and cloudlet technologies are examples of distributed technologies.

Almost the same management strategy has been suggested for the decentralized-to-centralized and distributed-to-centralized ICT architectures in smart cities, as shown in the right-hand side of Fig. 4. In both ICT architectures, all city-data may be saved and processed locally close to the city-data sources and citizens. Also, city-data can be transferred into a centralized cloud storage platform if it requires city managers and citizens' demands. This management solution coordinates resources, data/databases, software/services, and network communication and its cybersecurity issues from the edges of networks and nearby to the city-data sources to a centralized platform and most far away from the city-data sources.

The core difference between decentralized-to-centralized and distributed-to-centralized ICT architectures is the relation between edge nodes, middleware, and cloud computing platforms. In decentralized-to-centralized ICT (DC2C-ICT) architecture, the edge IoT devices may not interact directly with each other at the edge of networks without moving and accessing the Cloud platform. Distributed-to-centralized (D2C-ICT) architecture makes it possible for edge IoT devices to communicate with each other at the edge of networks without moving and accessing the Cloud platform. Linking distributed and centralized technologies in smart city integrated architecture have multiple benefits [11, 12]. Some examples are limiting data traffic in network communication and their related latencies and bandwidth and raising data privacy and cybersecurity matters.

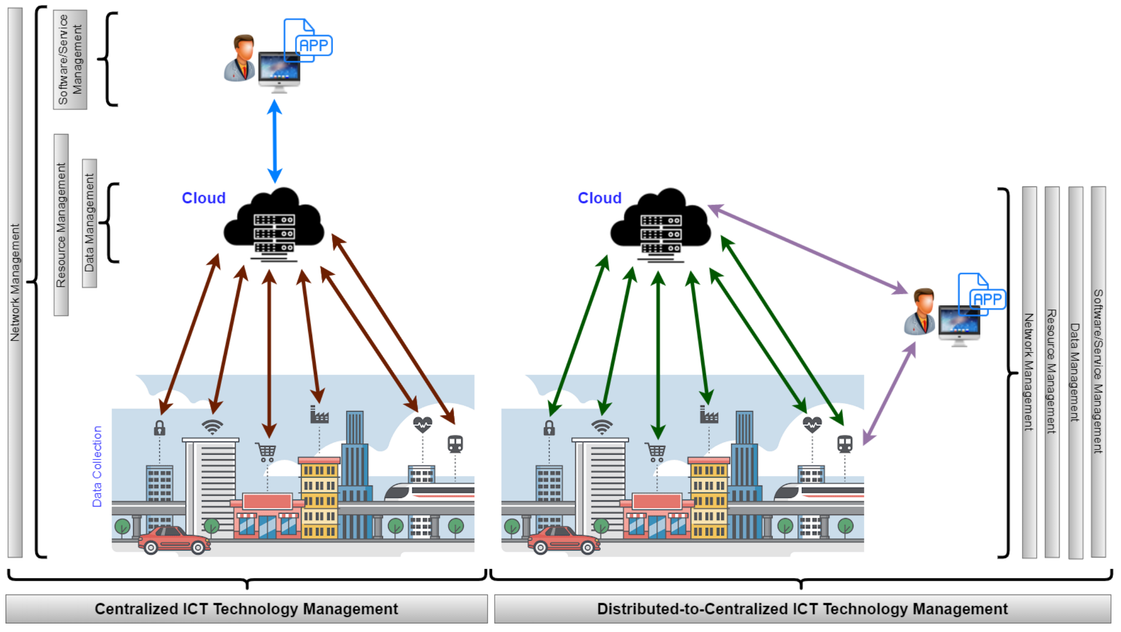

Software/Service Management
Resource Management
Data Management
Network Management
Data Collection
APP
Cloud
Cloud
APP
Network Management
Resource Management
Data Management
Software/Service Management
Centralized ICT Technology Management
Distributed-to-Centralized ICT Technology Management

Figure 4 Proposal for the ICT Reference Architecture for Smart Cities [13]

2.3 Our proposed ICT architecture in smart cities

A smart sustainable city refers to a city that requires the ICT infrastructure in an adaptable, reliable, scalable, accessible, secure, safe, and resilient manner [14]. Also, one of ICT architecture's main objectives is to provide facilities for building efficient software services through large-scale ICT networks of the smart city. So, some essential functions to build efficient software services in large-scale ICT networks of the smart city are interoperability, GDPR, mobility and location-awareness, real-time data access, scalability, reliability/high availability, and energy efficiency [15]. Both the smart sustainable city's objectives and software services requirements highlight the necessity of designing large-scale ICT networks for smart cities. Below, we describe our experience to design large-scale ICT networks based on edge-to-cloud orchestration.

In the beginning of our studies for designing a large-scale ICT architecture, a decentralized-to-centralized ICT (DC2C-ICT) architecture was proposed using Fog-to-Cloud technologies [12]. We designed our proposal for DC2C-ICT architecture in a smart city that is able to manage sensors and physical IoT data sources. Next, more challenges and requirements were suggested to our DC2C-ICT architecture for smart cities, as noted below:

- All physical and non-physical city-data sources in the city may need to be managed through the proposed ICT architecture;
- All city-data, and non-city data may be required for contributing to enhancing citizen life.

Consequently, we designed the D2C-ICT architecture using different distributed and centralized technologies, including Fog, cloudlet, and Cloud technologies. This architecture will help to solve the above challenges and requirements and bridge the gap, combining IoT and other data inside the city.

This section is written into three main subsections. First, we discuss the DC2C-ICT architecture in smart cities. Second, we explain the D2C-ICT architecture in smart cities. Finally, we describe the hybrid ICT architecture.

2.3.1 DC2C-ICT Architecture

Many physical and non-physical data sources are spread across the city in different “Fog-Areas” as shown in Fig. 5. A Fog-Area is a specific area in the city that is coordinated directly by the related Fog-Leader device(s). In our proposal for DC2C-ICT architecture [16, 17], the Fog-Leader devices can be routers, switches, gateways, or raspberry-pis. A Fog-Leader device(s) of the Fog-Area can organize processing and storage locally at the network's edge. This Fog-Leader may handle all related processing and storage tasks by itself or create a joined processing and storage facility by utilizing the potential of other accessible city-data nodes in the Fog-Area within the Things/Fog layers.

Our DC2C-ICT architecture is designed with three architecture layers, particularly Things-Layer, Fog-Layer, and Cloud. In this scenario, all Fog-Leader devices can communicate and get access to the Cloud platform. The Cloud platform is responsible for sending all instructions and updates to the lower layers. A Fog-Leader device may communicate with other Fog-Leader devices in another Fog-Area in the city by communicating and accessing through the Cloud platform.

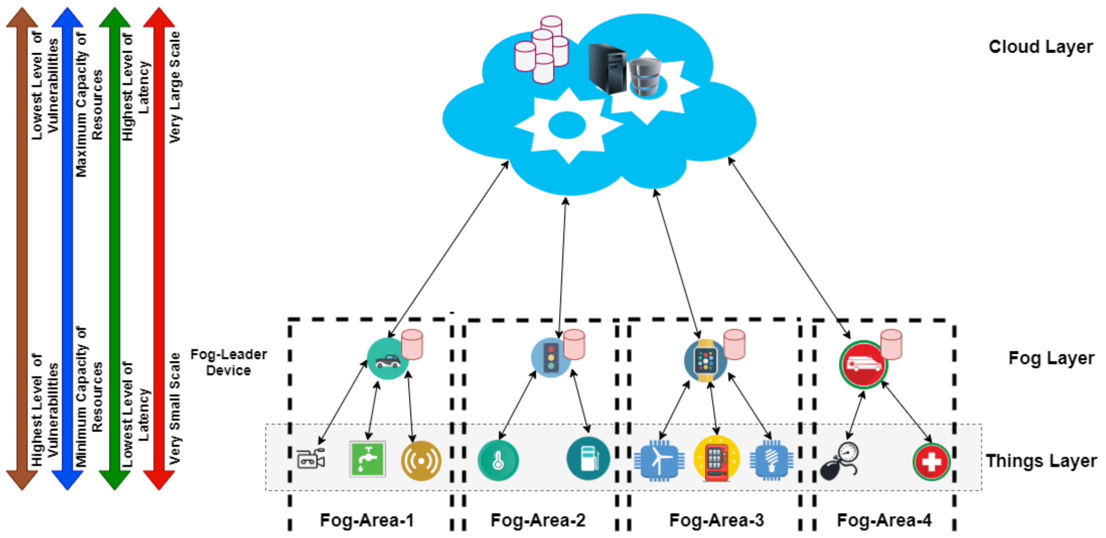


Figure 5 Our proposed DC2C-ICT architecture for smart cities

<u>2.3.2 D2C-ICT Architecture</u>

Our proposed D2C-ICT architecture is depicted in Fig. 6 by using Fog, cloudlet, and Cloud technologies. This architecture has four layers, as described below from bottom layer to top layer:

- Things Layer: This layer is in the bottom layer of Fig. 6 and is the layer closest to citizens. Physical and non-physical city-data sources in large-scale IoT networks of a smart city are available in this layer. Depending on their city location, the city-data source may be located in a different Fog-Area of the city. This layer is mainly responsible for data generation in smart cities through different IoT and IoT devices across and within the city. All city-data sources may have limited computational and storage capacity for their own data management requirements and share resources with other neighbours for joint processing and storage tasks.
- Fog Layer: The Fog-Layer is the second bottom layer of Fig. 6 and the second closest layer to the city's physical data sources. This layer includes switches, routers, and gateways to handle data management of obtained city-data from the bottom layer. This layer can provide a higher level of processing and storage compared to the bottom layer. Fog computing technologies implement local data processing and storage facilities in a city instead of sending them to the Cloud. Also, physical, and everywhere connected city-data sources create real-time city-data in this layer. Also, this layer is even capable of building private and critical city-services for citizens (Fog-as-Service).
- cloudlet layer: This layer is in between the Fog and Cloud-Layers and in the same city as city-data sources. It is public for the city and city customers, but it is private for the global cloud customers. This layer may add to the city's external sources, such as the server in the box (cloudlet). cloudlet computing technologies make a higher level of processing and storage in the city to perform joint processing and storage tasks by adding external physical servers in a city. Plus, these external physical servers and devices can be installed in a dynamic or fixed position of a city. All physical and non-physical city-data sources be combined in this layer anywhere in a city. The last-recent city-data is ready in this layer. This layer may create local and private city-services (cloudlet-as-Service) by using all city-data sources. This layer also creates communication between city-data sources at the edge of networks. This communication may improve city-data nodes' participation in a city without transferring and accessing Cloud technologies.

At cloudlet-Layer, we create a middleware platform called an "Integrated and Intelligent Control and Monitoring of IoT (I2CM-IoT)" box. This I2CM-IoT middleware platform may coordinate all city resources and accessing Cloud resources from all city scale to a centralized place. This middleware platform may create data discovery and data monitoring mechanisms for all city scales.

- Cloud layer: The top position of Fig. 6 is Cloud-Layer. Cloud technologies may not be positioned in the same city/country/continent of city-data sources. Cloud-Layer has diverse and nearly endless technology resources that can organize almost all IoT and ICT obligations in a centralized platform [9, 10]. All city-data sources may be stored in the Cloud platform or reachable through the below layers. Non-city data sources may also be reachable at a centralized Cloud storage platform. So, all historical city-data is accessible in this layer. This layer is useful for historical and public services (Cloud-as-Service).

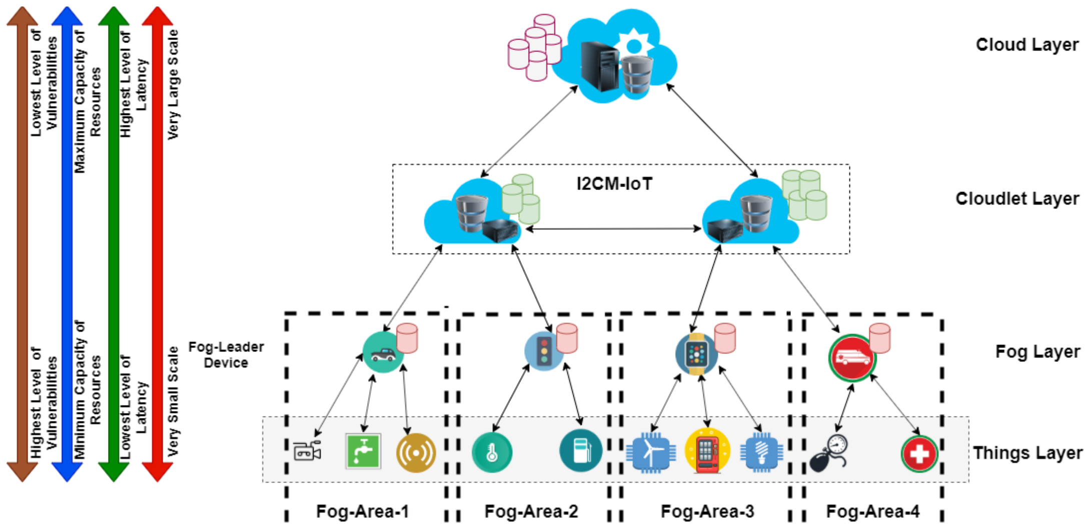


Figure 6 Our proposed D2C-ICT architecture in the smart city

"Time" and "Location" are two primary axes on our proposed D2C-ICT architecture, as shown in Fig. 6. Those axes describe our core concept about large-scale ICT management for IoT networks in smart cities. Those concepts are "data/database management," "resource management," "network communication management," and "service/software management," as described below:

- ***Data/Database Management:***

In general, "data management" is an organizational process that comprises acquiring, validating, storing, protecting, and processing required data to ensure the accessibility, reliability, and timeliness of the data for its customers. Data management intends to support data stakeholders to optimize data usage within the policy and regulation ranges to create decisions and enhance its benefit.

In general, the term "database" defines a collection of data that can be accessed, modified, managed, controlled, and organized for numerous data-processing tasks. Database management refers to organize who can access a database and how they can make corrections. It is necessary for coordinating the security and integrity of data.

Relevant concepts of data and database management in smart cities are explained below:

1-Data Types: In terms of the time of the produced data in a city [11], three types of city-data are available from Fog-Layer to Cloud-Layer in smart cities, "real-time," "last-recent," and "historical," as explained below:

- Real-time data: Real-time city-data at Things-Layer is delivered from a place nearby to the citizens by physical IoT data sources. The real-time city-data is the item that is generated and used directly, fundamentally in critical shallow latency software services in smart cities.
- Historical data: Data is assumed historical city-data at the cloudlet/Cloud layer if added and kept on the data repositories platform for future use. In this case, historical city-data can be recognized as the farthest place from the citizens and physical data sources in the city. Hence, accessing city-data from cloud computing technologies has a high latency level compared to the below layers.

- Last-recent data: The last-recent city-data at Fog-Layer is not as close to the citizens and physical data sources as the real-time data are. The last-recent city-data is also not far away from the citizens' citizens and physical data sources as the historical city-data are.

In terms of the city-data sources, physical and non-physical data sources generate diverse data types in a city, as specified below.

- Portal-Sensor data: This city-data type produces with portal sensor devices in a city. So, these data sources can move and be available in different city locations, e.g., vehicular and smartphone data. E.g., vehicular and smartphone data.
- City-Consumer data: These city-data types available at the cloudlet-Layer get data from many city-consumer databases, including city-hall databases and private and public company databases.
- Web data: These city-data types is available at Cloud-Layer obtain from various websites.

2-Data Flow and movement: Three different data flow and movement appear in the multilevel of Edge-to-Cloud orchestration in smart cities, bottom-up, top-down, and cross-cutting, as explained below.

- Bottom-up approach:
  - From Fog to cloudlet: Fixed- and Portal-Sensor generates “real-time” city-data in the city. These city-data types may move from bottom to upper layers for additional data processing and permanent/temporary data storage reasons through the city-update mechanism. If the real-time data is saved at Fog-Layer, just the references of information and abstracted form will be shifted and addressed to the higher layers for extra data discovery.
  - From cloudlet to Cloud: all city-data can be shifted from cloudlet-Layer's storage platform to the Cloud-Layer for additional data processing and permanent/temporary data storage reasons. If the last-recent data saved permanently at the cloudlet-Layer, just the references of information and description of this data would be shared to the upper layers' address table for extra data discovery.
- Top-down approach: a reference of “Historical data” available in the Cloud storage platform could be shared from Cloud or cloudlet-Layer to the Fog layer. In this case, seeking the historical data from the network's edge can be arranged speedily.
- Cross-cutting approach: The cloudlet layer's accessible data can be accessed and communicate with another cloudlet platform in a city.

3-Data Aggregation: In general, data aggregation builds some processing abilities for gathering, reducing, mixing, or presenting summary information [12]. Reducing the generated data amounts is the core purpose of data aggregation techniques. Various techniques can be applied to achieve the purpose of data aggregation, e.g., data combination, data redundancy elimination, data compression, or bandwidth reduction.
Our proposal for smart cities is the multilevel techniques of data aggregation, and data compression can be applied through different layers of Edge to Cloud technologies [12]. We demonstrated the multilevel data aggregation and compression techniques to Barcelona smart city in our use case. The result [12] confirms our proposal's efficiency using Edge-to-Cloud orchestration compared to Cloud technologies.

4-Data Storage: In our smart cities' proposal, we illustrated how the city-data might be moved from local data repositories at Fog-Layer to global data repositories platform at the Cloud layer [18]. This multilevel update mechanism may grow the network traffic and storage performance [18].
5-Data Discovery: In general, the process of searching, finding, and taking data from diverse databases is data discovery.

We apply a particular cost model (MAUT) for data discovery through Edge-to-Cloud layers in smart cities [19]. The cost model techniques are useful for filtering and calculating a ranking among all viable choices based on the query's criteria.

6-Data Management: Our proposal for distributed-to-centralized data management (D2C-DM) architecture is presented in [11]. The purpose of the D2C-DM architecture is to handle the city-data management from data creation to consumption in multiple levels of the D2C-ICT architecture (Data LifeCycle [20] model) and mix city-data with non-city data.
A Smart City Comprehensive Data LifeCycle (SCC-DLC) model [21] describe the core idea of data management in a city. The SCC-DLC model's three main blocks are "Data Acquisition," "Data Processing," and "Data Preservation." Each block contains a set of separated phases as described shortly below and in more in-depth details in [21]:

- The Data Acquisition block can implement diverse data operations and phases for the produced city-data, e.g., data collection, data filtering, data quality assurance, and data description phases.
- The Data Processing block is responsible for controlling the many activities connected with the data process and data analysis phases.
- The Data Preservation block refers to varying data actions and phases, including data classification, data archive, and data dissemination.

7-Database Management: Database management is one of the most critical features that we require to be discussed. As the various sorts of a large number of computing devices are working, and several organizations are participating, it is challenging to handle the generated data. Due to the data privacy policy, many organizations and system users may not find it possible to distribute their private data. A huge number of small connected computing devices are part of the Things-Layer of the smart city. Those devices can produce a large amount of data within the system. Also, different organizations are utilizing different sorts of database framework to hold and manage their organizational data. So, altogether, it is challenging to handle those data near the network's edge. In our solution, we attached the cloudlet-Layer between of the Cloud- and the Fog-Layer. The cloudlet-Layer provides the cloud abilities close to the edge, including fast data processing and analysis in the city.

- ***Resource Management:***

The broad diversification among the system resources presents the challenge for the system administrator to efficiently control the participating devices in a large-scale ICT-based smart computing scenario in smart cities. The massive diversity among the system services creates difficulties to any Large-Scale ICT-based smart computing paradigm. With our proposed D2C-ICT architecture, we can discuss those difficulties and efficiently handle the system resources to decrease undesirable resource consumption without compromising quality of service. The resource management mechanism is essentially incorporating a few relevant operations or steps in our D2C-ICT architecture. By performing all the below steps, some task(s) can be executed for achieving some output(s) for completing some service(s) request.

a) Service(s)/Job(s)/Task(s) is identified, and one get their required information;
b) Gather the possible system resource(s) information;
c) Choose the most suitable matching resource(s) for meeting the requirements of the service(s)/ job(s) / task(s);
d) Allocate the best-fitted resource(s) for performing some task(s) to fulfill some job(s) for implementing the service(s) among the system subscriber;

e) Monitor the operating resource(s) to trace the resource consumption information to develop the future matching score.

• ***Network Communication Management:***
In general, this points to a wide range of functions linked with the management and control of network communication. Network management functions include management of fault, configuration, accounting, performance, and security. Some functions are real-time; however, some others are administrative.
In our proposed ICT architecture in smart cities, we first suggest addressing and routing data sources' mechanisms from Edge-to-Cloud orchestration in smart cities. We apply the principle of the content delivery network (CDN) and information-centric networking (ICN). Second, we represent multiple controllers through multilevel layers of Edge-to-Cloud orchestration by adopting software-defined networking (SDN) technology. More details can be found in the following sections.

• ***Service/Software Management:***
In general, it relates to designing, delivering, managing, and improving the ICT services that service stakeholders perform to facilitate their end-users. A service/software management plan can assist in formalizing a set of structures and intentions that ensure that the city software is available and reusable for citizens.
In our smart cities' proposal, the city-data is saved and can be accessible wherever from city data storage to the Cloud storage platform [18]. Accordingly, various services can be launched into multiple computing technologies from Edge to Cloud, such as Fog-as-Service, cloudlet-as-Service, and Cloud-as-Service. These multilevel service layers are advantageous for local and restricted city-data that may not be possible to share on a higher level in the architecture.

2.3.3 Hybrid Architecture

Based on your business and demand for available centralized and distributed technologies in your city, you have an option to design your ICT architecture. Some example of hybrid architecture is depicted below:

- In Fig. 7, Fog-Area is divided into city-data sources types and categories in the smart cities. In this Figure example, all the camera surveillance city-data will move directly to Cloud technologies; that is how centralized ICT architecture is designed, as we discussed above. In this case, maybe because of the huge amount of this type of city-data may be challenging to store in the local data repositories and analyze them through distributed technologies.
- All energy and grid city-data will also follow the F2c2C-ICT architecture and move data through different distributed to centralized technologies. For instance, a reason for this design for the energy and grid city-data could be the high level of data privacy for these data types. It indicates that the citizens and companies may not like to transfer and share their own city-data types to the upper layers.
- All health-care city-data may be manage through decentralized-to-centralized ICT architecture with the assistance of Fog and Cloud technologies. For instance, this could be a reason for the real-time application and requirements of these business domains for the citizens. Also, they need to send some part of data or data models to the Cloud technologies for sharing to others for future requirements in the city and another city, e.g., building digital-twins' requirements in the Cloud technologies.

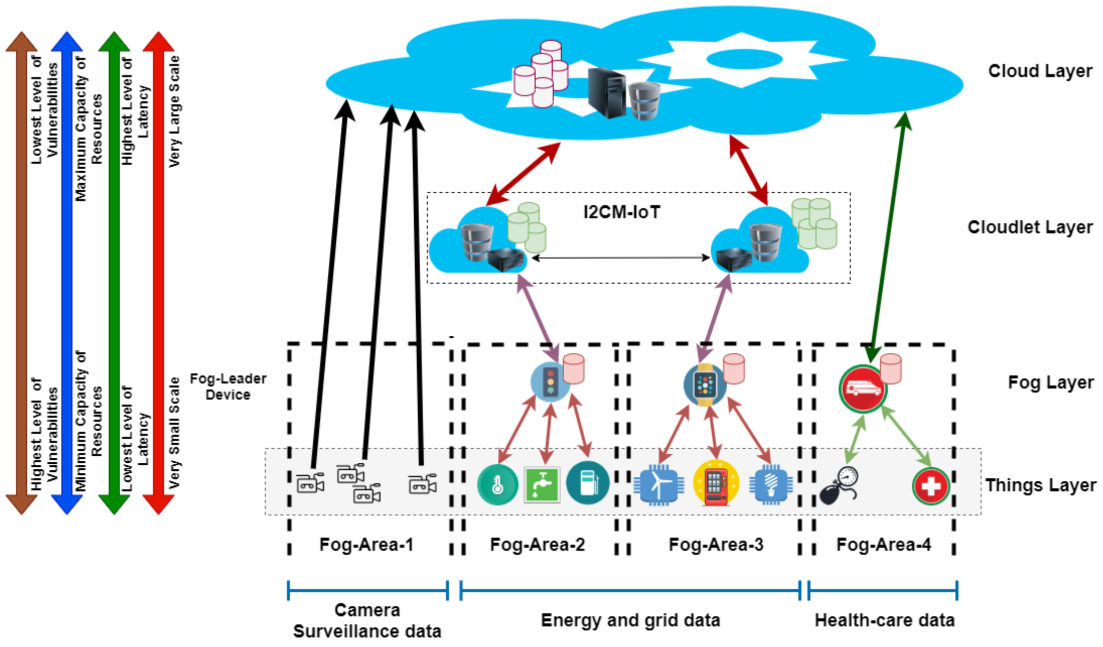


Figure 7 Our proposed hybrid ICT architecture in the smart city based on data-source formats

- As we mentioned above, the hybrid architecture is quite flexible to design concerning the citizens and business requirements. Fig. 8 shows that each Fog-Area that is a particular district of the city may need to be managed by different ICT architecture strategies, as explained below.
    - The left-hand side of Fog-Areas (Fog-Area-1) may be designed through centralized ICT architecture. Maybe the reason is the available technologies and/or the inhabitants' requirements in this city district.
    - The mid-hand side of Fog-Areas (Fog-Area-2 and Fog-Area-3) may implement by D2c2C-ICT architecture. The reason might be the data privacy and particular security and GDPR protection as requested by inhabitants and business demands in this city.
    - The right-hand side of Fog-Areas (Fog-Area-4) may structure within F2C-ICT architecture. The reason might be the availability of external city-data sources, for example, server or cloudlet technologies, in this city district.

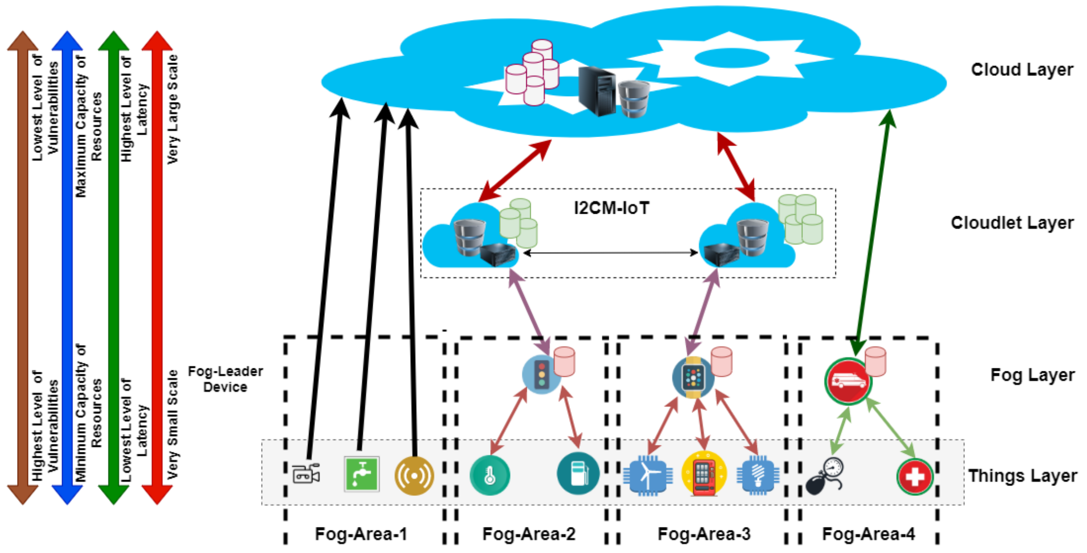


Figure 8 Our proposed hybrid architecture in the smart city based on city districts

3 Data Management

In this section, we focus on data/database management as one of the main concepts of large-scale ICT management in smart cities, as discussed in the previous section.

This section is separated into two main subsections, including data management in context and data management architectures in smart cities.

3.1 Data management in Big Data Management context

Data are one of the most precious resources in the smart city. City-data provides the ability for a city to be smart, agile, and creative. Smart cities build much city-data from various sources in different formats, from IoT devices to city-consumer data, third-party applications, and social media. Numerous studies on data management and analysis are done in smart cities, from Relational Databases Management Systems (RDBMS) to the Extract-Transform-Load (ETL) process [20]. Big Data environments' evolution also add extra challenges to the traditional data management and analysis systems [20, 22].
The entire life cycle of data management strategy, including data collection, data storage, and data processing, is complicated [23, 24]. The primary goal of data management is to guarantee easy and safe access to data sources and their associated repositories to identify additional value through complicated computing and analytical processes utilizing the sources. Adequate data management and organization systems can be considered an essential topic for effective value generation. Also, Big Data creates remarkable opportunities and difficulties for data management. In short, the enormous opportunities of Big Data include its economic influence [20, 25]. Big Data is characterized by the d 5Vs challenges, usually named Volume (huge volume of data), Variety (various data formats), Velocity (rapid generation and processing of data), Value (huge value but very low density), and Veracity (quality of data).

3.2 Data management architectures in smart cities

This subsection is sorted into three subsubsections—first we will discuss a CDM architecture in smart cities. Second, DC2C-DM architecture will be explained through the Fog-to-Cloud data management (F2C-DM) idea. Third, D2C-DM architecture will be described, including cloudlet-to-Cloud data management (c2C-DM) and Fog-to-cloudlet-to-Cloud (F2c2C-DM) architecture.

3.2.1 The CDM architecture

In smart cities, the CDM architecture applies to data management in a centralized ICT architecture [9]. This is based on that all city data sources can be organized and managed in a central place. The central place is then used for accumulating all city-data and contributing all the city-data to non-city data for further use. Besides, a central place performs and handles all data management procedures, including data processing, storage, analytics, aggregation, etc. As an illustration, Fig. 9 shows a CDM architecture in the smart city based on cloud computing technologies [9]. The ICT architecture has four layers: physical, network, cloud, and application. With a focus on data management procedures and strategies, the third layer is the second top layer of Fig. 9 that performs all data management procedures, such as processing, storage, and analyzing all raw city-data and transforming them into meaningful information to be used in further software services and application requirements. As a bottom layer of Fig. 9, the physical layer includes all physical and non-physical data sources within and across the city to capture all city activities. So, this layer is only responsible for the data collection phase and data management procedures.
In another example [26], as shown in Fig. 10, the author explicitly mentioned that obtained city-data would be sent to the cloud computing technologies by different internet connections and then in the cloud computing technologies, different data management procedures will occur, including data collection/aggregation, filtering, classification, preprocessing, and all issues related to ML and decision making.

These examples highlight the main challenges of CDM architecture. The data collection occurred in a city somewhere close to the city-data sources, but all related data management procedures and strategies will happen far away from city-data sources [26].

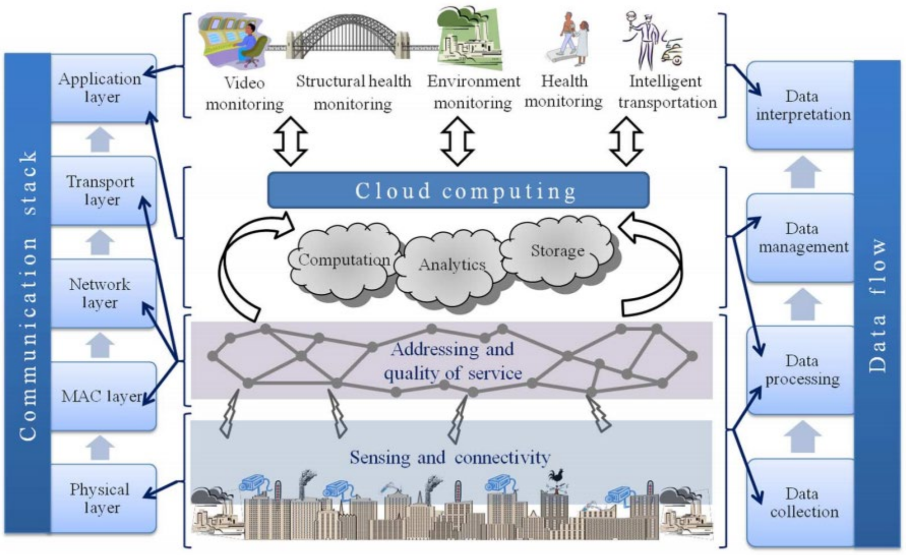


Figure 9 CDM architecture through centralized ICT architecture in smart cities [9]

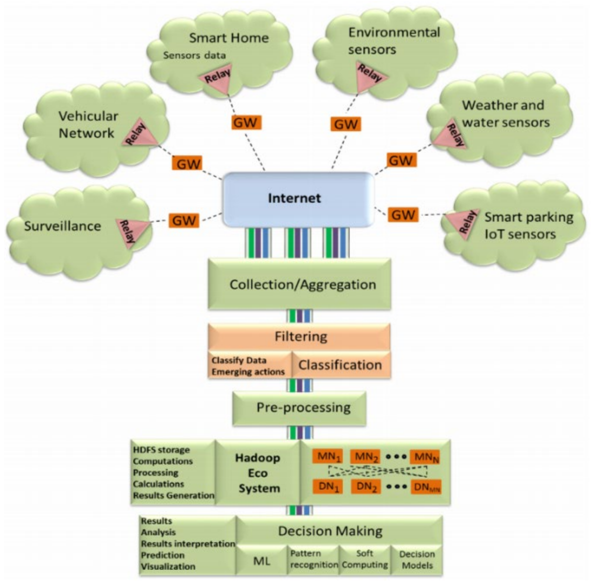


Figure 10 Data management strategies and steps in the Cloud platform [26]

3.2.2 The DC2C-DM Architecture

The idea of decentralized-to-centralized data management (DC2C-DM) architecture can be demonstrated across different layers of D2C-ICT architecture. In our proposed DC2C-DM architecture [12, 17, 18, 25, 27], the Fog computing technologies [28, 29] are used, as described below:

The F2C-DM Architecture

Fog computing technologies have been developed by among others by Cisco [30]. The fog computing idea shows that the physical devices, such as routers, switches, and raspberry-pi, can contribute to IoT devices and data sources at the edge of networks. These contributions can be beneficial for data and resource management requirements, [31-33]. By this management strategy, city-data it may not be required to push all data to the central Cloud [12, 27, 34, 35]. Also, DC2C-DM architecture uses the Cloud computing platform for a sophisticated level of storage and processing. The orchestration of Fog and Cloud technologies makes facilities access real-time data close to the data sources, which can help develop the critical applications and processing requirements in a city [12, 27, 34, 35].

In our proposal [12, 27, 34, 35], the Fog-to-Cloud data management (F2C-DM) architectures are mentioned by a variable number of architectural layers from Fog to Cloud that can be matched with the business, citizens, and data model requirements of the city. E.g., we draw three layers for DC2C-ICT architecture, as shown in Fig. 11. The Fog-Layer-1 includes a set of IoT nodes at the edge of the network to perform the first level of limited processing and storage. The Fog-Layer-2 consists of different Fog-Leaders used for the second higher computing and storage level by joining their own resource and other under layer resources at the network's edge concerning the city's available devices' capabilities. Finally, all nodes and leaders of Fog-Layer-1 and Fog-Layer-2 must connect to the Cloud technologies for organizing the top management instructions, such as updating policies and data management instructions. The cloud layer is the top-most level in F2C-DM architecture. The most powerful computing and storage equipment is used in this layer because the Cloud platform has nearly countless resources, although latency is higher.

As a result, the main idea of the DC2C-DM is to coordinate data management from data creation to consumption, which is fitted to the base idea of the Data LifeCycle model [23, 24]. As shown in Fig. 12, our proposed DLC model for smart cities [34, 49] designed as a Smart City Comprehensive Data LifeCycle (SCC-DLC) model to determine data management strategies idea through their entire life cycles from different blocks of data acquisition to data preservation and processing and their related phases. This involves other major activities related to data quality and data security, also relevant in a city, as presented in Fig. 12. The SCC-DLC model's main organization is described with three blocks, including Data Acquisition, Data Processing, and Data Preservation, and each block comprises a set of varying phases, as described in more detail in [12, 27, 34, 35].

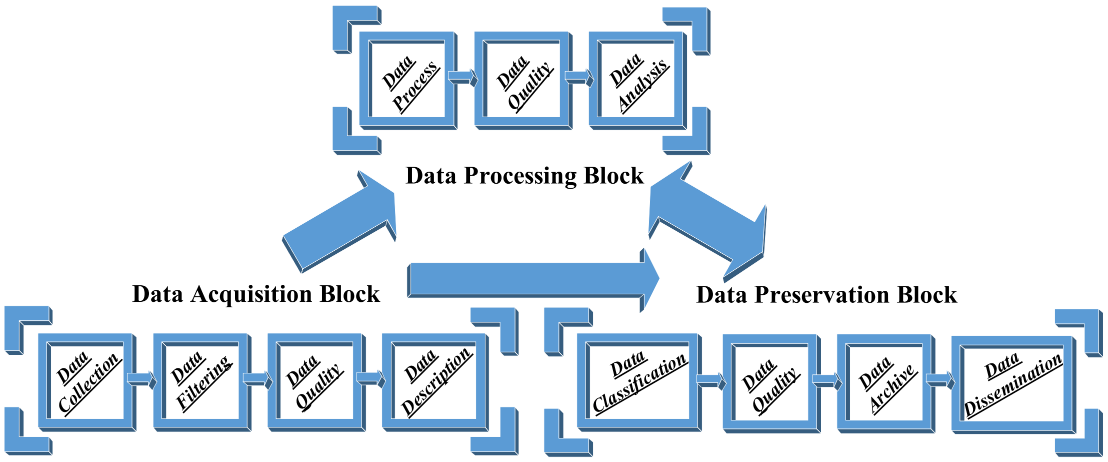


Figure 11 The DLC model [25, 36]

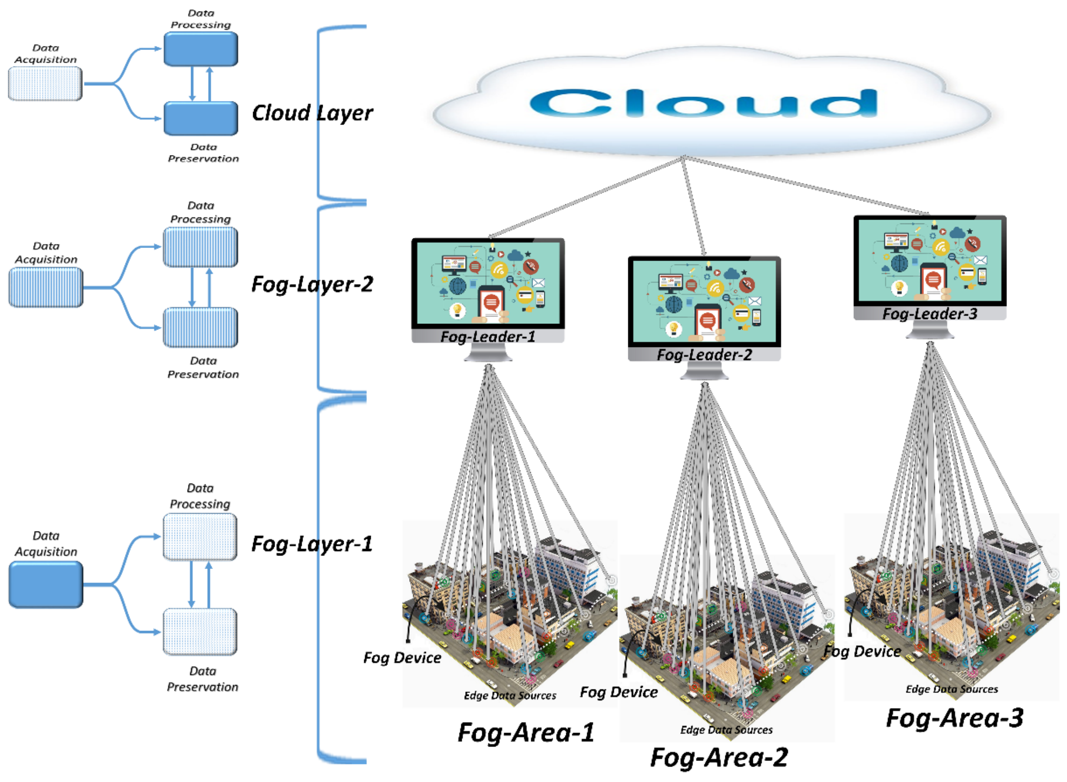


Figure 12 The F2C-DM architecture in smart cities [12]

3.2.3 The D2C-DM Architecture

The idea of D2C-DM architecture can be demonstrated across different layers of D2C-ICT architecture by different technologies, such as cloudlets. In our proposed DC2C-DM architecture [12, 17, 18, 25, 27], the Fog computing technologies [28, 29] are used for data management architecture in smart cities. Like DC2C-DM, D2C-DM also followed the DLC model's principal idea for data management in smart cities. It means data management will handle data from data creation to data consumption.
This subsubsection is divided into two subsections—first, cloudlet-to-Cloud data management (c2C-DM) will be presented. Second, we will describe the idea of Fog-to-cloudlet-to-Cloud data management (F2c2C-DM).

The c2C-DM Architecture

Cloudlet technologies have been mentioned by Carnegie Mellon University (CMU) [30]. Fog and cloudlet technologies try to address the same objectives. Though, the core distinction is the capacity of resources of Fog and cloudlet technologies. Fog technologies are using the potential of the smart cities' data sources, such as routers and raspberry-pi, but the cloudlet technologies serve as the data center in the city close to the network's edge. So, the idea above addresses the Cloud co-operations closer to the citizens and city-data sources [30].
Like F2C-DM, cloudlet-to-Cloud data management (c2C-DM) architecture may be offered by different architectural layers, as shown in Fig. 13. c2C-DM provides facility-to-direct communication with cloudlets technologies at the edge of networks. c2C-DM brings the management levels in a city. It means all instructions and policy mechanisms can be sent from cloudlets to Fog layers and IoT nodes at the edge of networks. If it is relevant, cloudlets can communicate and get help from Cloud technologies to update their instructions and update mechanisms. Like the F2C-DM architecture, data management strategies, such as processing and storage, can occur from bottom to top layers with different resource

capacities. The lowest capacities are in the edge of the network at the bottom layer, and the most powerful and almost unlimited capacities are in the Cloud platform, as shown in the arrow of the left-hand side of Fig. 13.

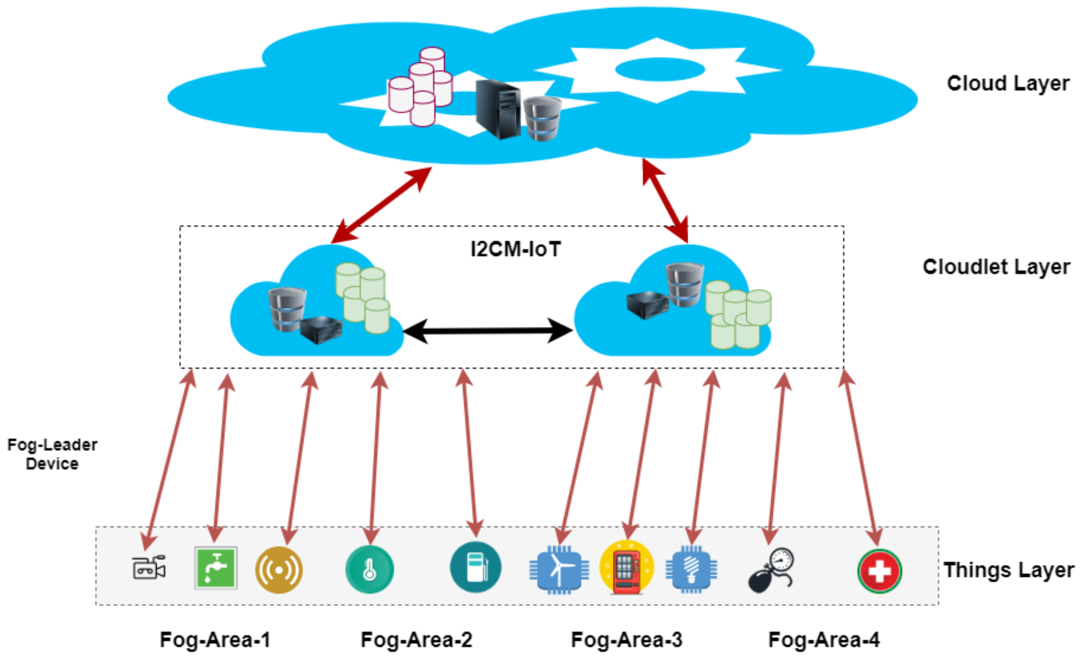


Figure 13 The c2C-DM architecture for Smart Cities

3.2.4 The F2c2C-DM Architecture

Diverse technologies support our F2c2C-DM architecture [11] at the multi-level of D2C-ICT architecture, as shown in Fig. 14. Those technologies are Fog and cloudlet as distributed technologies and Cloud as a centralized technology. This means we combine all advantages CDM, DC2C, and c2C-DM architectures in a unified architecture. This makes several benefits for data management in smart cities, as shown below:

1. Organize all physical city-data sources by helping Fog technologies and non-city data sources by supporting cloudlet technologies in a city as close as possible to the network's edge.
2. Use and share all city resources potentials for city-data management from distributed to centralized schema, including data processing and storage.
3. Perform multi-level data management techniques to organize all obtained city-data from distributed to centralized. This could be aligned to data privacy and city and user requirements across and within the city.
4. Data management strategy can be worked separately in a city by helping with distributed technologies without Cloud technologies' assistance. However, if necessary, the Cloud computing technologies are reachable somewhere out of the city in most cases.
5.

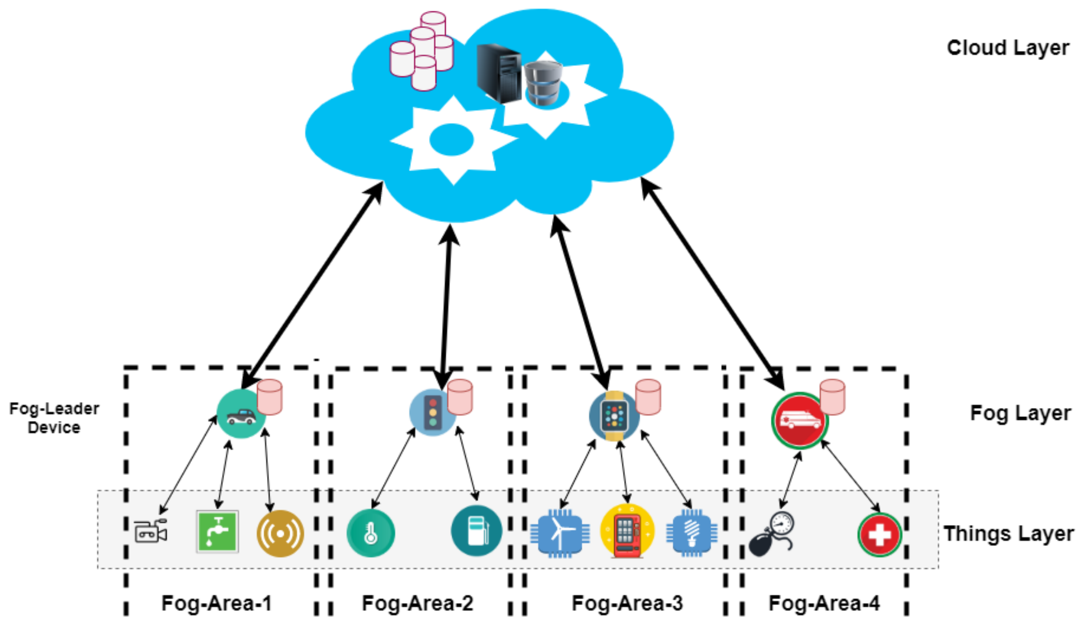


Figure 14 The F2c2C-DM Architecture for Smart Cities

4 A Proposed ICT Architecture and its Data Management Architecture for ZEN Centre

This section is organized into five main subsections, as shown below.

4.1 ZEN Background

There are seven pilot areas of neighborhoods implementing the zero-emission principles in ZEN [1, 37]. Table. 1 gives an overview of the pilots. As can be seen, they vary substantially in their characteristics and will cover a range of scenarios and use cases and provide more challenges in developing a consistent data architecture.

The projects range from small housing and public buildings over small and large campuses to large functional city areas and whole neighborhoods. Further, they consist of a mix of redevelopment and new construction.

This also influences the type of buildings, infrastructure, and entities found in the geographic pilot areas.

Table 1 Overview of the seven ZEN pilot projects [1]

	City population (1.1.16)	Project owner	Area size in m^2	Planned/ Current Function	Construction	Status/Phase
Elverum - Ydalir	14 794	Public (Municipality)	430 000	Residential area with school and kindergarten (planned)	New construction, 1 000 dwellings (ca. 100 000 m^2), a school and a kindergarten	Planning
Oslo - Furuset	652 940	Public (Municipality)	870 000	Multi-functional sub centre with 1 400 dwellings and 3 800 inhabitants, 213 100 m^2 (existing)	Retro-fitting/upgrading and new construction, 1 700 – 2300 dwellings and 2 000 – 3400 work places (up to 160000 m^2)	Planning and Construction
Bergen - ZVB	252 772	Private (Developer)	378 000	Residential area with a kindergarten and additional services (planned)	New construction, 720 dwellings (92 000 m^2), a kindergarten and additional service functions	Planning
Trondheim – NTNU Campus	177 617	Public (NTNU/ Municipality)	339 031	University Campus (existing)	Retro-fitting and new construction (ca. 136 000 m^2)	Planning
Steinkjer – NRK	12 466	Public (Municipality)	11 113	Kindergarten and dwellings (planned)	Re-use and new construction of 10-12 dwellings	Planning
Evenstad - Campus	2 623 (Municipality)	Public (University)	61 000	University Campus (existing)	Building stock in use: 10 000 m^{2,} no further construction planned	Operation
Bodø - NyBy	40 209	Public (Municipality)	3 400 000	Multifunctional city centre extension with residential and business areas (planned)	Re-use and new construction, 2 800 dwellings in the first construction stage	Planning

4.2 Conceptual Levels of ZEN Study

We mapped this paper's idea [38, 39] for our conceptual level of the ZEN study. So, the conceptual levels model around a city, neighborhoods, and buildings is depicted in Fig. 15, which presents three primary levels: micro, meso, and macro. Fig. 15 showed that the top of Fig. 15 is the macro level and consists of all city's neighborhoods with a very large model of a city. The mid-level considers meso-level of the city, and all neighborhoods are included, such as all buildings, houses, streets. The bottom-level is the micro-level of the city that is all building components.

Fig. 15 presents the neighborhood at the meso-level within the micro and macro levels of buildings and cities. The ZEN objective concentrates firmly on the neighborhood level.

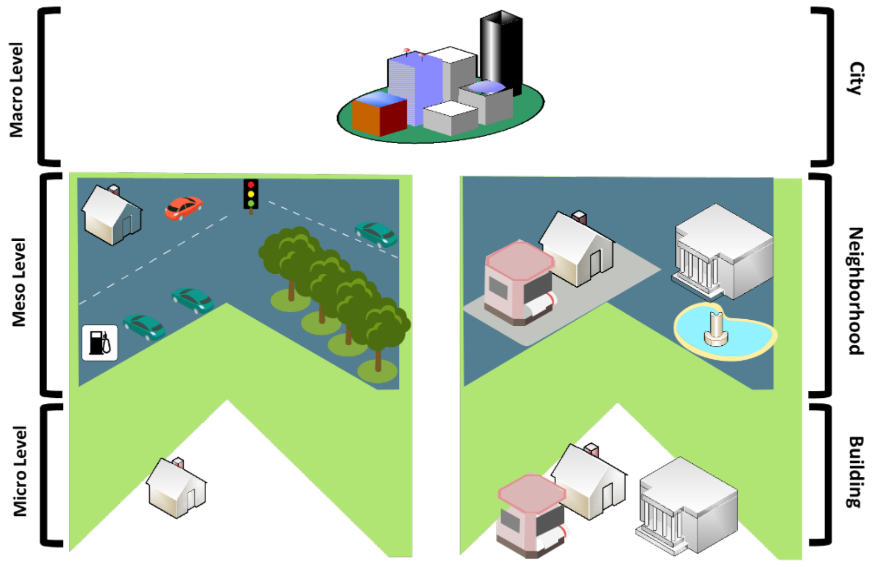


Figure 15 Conceptual levels around a city and their neighborhoods and buildings

4.3 ZEN Data Types

This subsection examines data and data flow and sharing needs along with a variety of dimensions for a thorough understanding of the ZEN project environment. Of further interest is the linking and relation between these types of data, their overlap, their use, data flow, and the sharing and reusing of it.
All different data types of ZEN center are categorized, as described below:

- Context data: Normally, all these different data types have been generated by IoT and sensors devices or come from external systems or repositories, such as weather, environmental conditions, energy systems, urban planning data, etc. This type of data supports the interpretation of results and other data but is not of interest solely on their own.
- Research data: These data types have been generated by third-party applications, such as simulation or planning data, or sensor streams. This data, which has been collected and used, can come from pilots or entities/installations/buildings within pilot areas prototypes from live building and energy data from buildings. Some examples of these data types are occupant behavior data, energy data, etc.
- KPI data: Planning Instruments for Smart Energy Communities (PI-SEC) is a Norwegian research project that aims at developing a toolkit for effectively planning both building project development (Bottom-up) and municipal development (Top-down) concerning energy issues [53].

The research project collected and examined KPIs from existing literature and determined 21 indicators that are important for smart sustainable cities. The goals that the KPIs contribute to are categorized in five areas:

- CO2 reduction;
- Increased use of renewable energy;
- Increased energy efficiency;
- Increased use of local energy sources;
- Green mobility.

To sum up, three data types are defined by ZEN center, including context, research, and KPI data. We matched these three ZEN data types with the ZEN conceptual models, as shown in Fig. 16. Therefore, first, the context data is relevant on all levels (including micro, meso, and macro). Second, the research data can exist on meso and macro levels. Lastly, in a similar way to the context data, the KPI data is potentially available in all levels (including micro, meso, and macro).

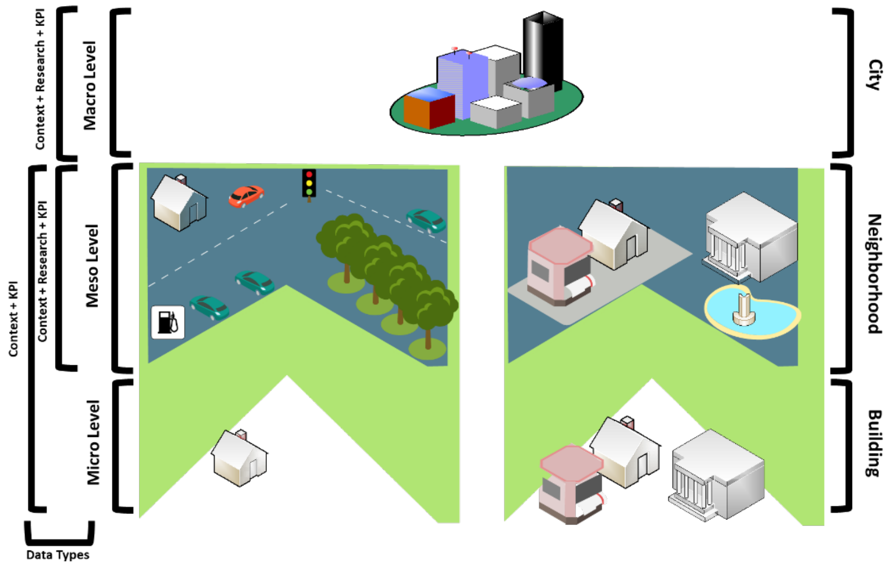


Figure 16 Data types in ZEN center through the ZEN conceptual models

Tailoring Smart City Data Types to ZEN Centre Data Types

As we discussed in the subsections 2.3.2 and 4.3 about data types in smart cities and ZEN center, on the one hand, there are three main different data types in the smart cities, including real-time data (physical data sources), last-recent data (non-physical data sources), and historical data (non-physical data sources). On the other hand, the ZEN center has three main distinct data types, including context, research, and KPI data. Therefore, Fig. 17 tailored the smart city and ZEN center data types to the conceptual models, as shown below:

- Micro Level: the real-time data produces near the end-users in the city by physical data sources (such as sensors). Also, the real-time data can be matched with the definition of the “context data,” “research data”, and “KPI data” for the ZEN center.
- Meso level: the meso layer covers both real-time (physical data sources on the neighborhood) and last-recent (non-physical data sources) as shown details below:
  - Real-time data: there are several physical data sources on the neighborhood (such as vehicular network, traffic lights, etc.) at the meso layer. Therefore, these data sources generate real-time data in the meso layer. Also, the real-time data can be matched with the “context data” and “KPI data” for the ZEN center.
  - Last-recent data: is located in the middle between real-time and historical data. Moreover, the last-recent data can be tailored to the “research data” and “KPI data.”
- Macro level: the macro level includes the historical data farther away to the city’s end-users. Moreover, the historical data can be tailored to the definition of the “research data” and “KPI data” for the ZEN center.

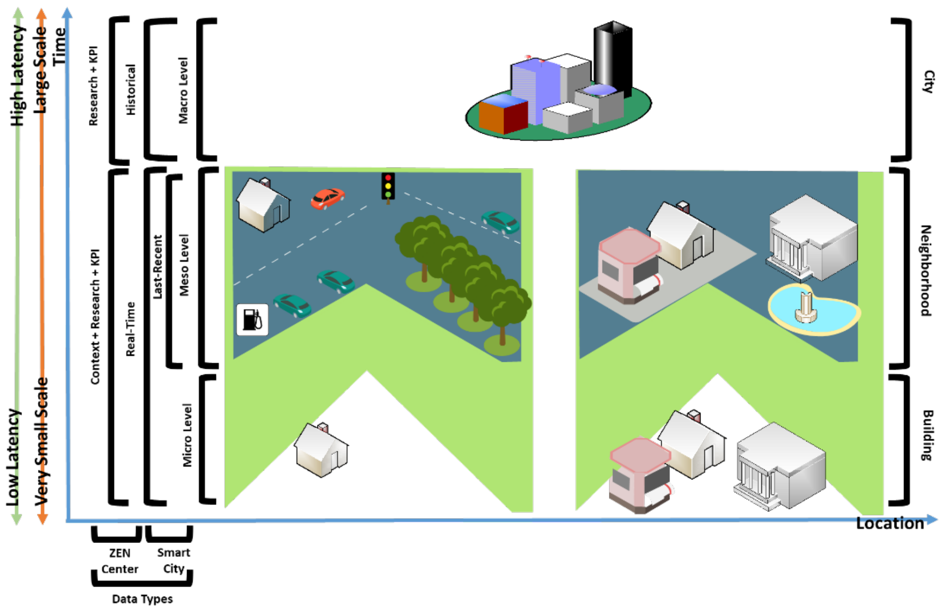


Figure 17 Different data types in smart cities and ZEN center

4.4 Interconnection of ZEN Definition, ZEN KPIs and ZEN Toolbox

There are two main boxes ("ZEN Definition Guideline" and "ZEN Toolbox Guideline") in Fig. 18 [40]. In addition, refer to the ZEN toolbox report [40], the interconnection of ZEN Definition, ZEN KPIs, and ZEN Toolbox is defined in the following:

- The "ZEN Definition Guideline" box aims to connect the ZEN definition and a set of KPIs. A set of KPIs (including GHG emissions, energy, power/load, mobility, economy, and spatial qualities) is defined to follow each ZEN criteria assessment.
- The "ZEN Toolbox Guideline" box creates a set of tools for the ZEN center to meet the "ZEN Definition Guideline" box requirements, including KPIs. There are different interconnections between ZEN KPIs and ZEN Toolbox. First, some of the tools can be assessed by a particular KPI. Second, multiple numbers of KPIs requirements can utilize other tools simultaneously.

The bottom side of Fig. 18 shows that the "ZEN definition guideline" and "ZEN Toolbox Guideline" can be connected to the ZEN pilot projects to discover and monitor their data requirements.

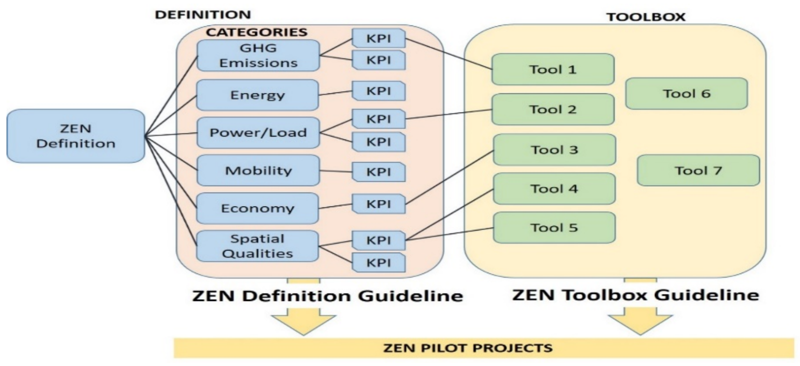


Figure 18 Interconnection of ZEN definition, KPIs and toolbox [40]

4.5 Large-Scale ICT management for ZEN center

We found our proposed D2C-ICT architecture including Fog, cloudlet, and Cloud technologies are quite relevant for the ZEN, and it is pilots based on the below reasons:

- Scale: The architecture must handle seven distinct pilots and cities in Norway.
- Flexibility: The architecture must be responsible for distinct ranges of policies in each cross-layer from edge to cloud technologies within city manager policies and ZEN business models requirements. Some policies example is checking data quality, applying data security, and updating frequency mechanisms.
- Complexities: Several management complexities may be addressed through ZEN requirements, e.g., varieties of data types and formats, mixing obtained data, etc.

To sum up, we proposed our F2c2C-ICT architecture for the ZEN center, as shown in Fig. 19. The ICT architecture has three different architectural layers from edge-to-cloud orchestration. The left-hand side of the architecture in Fig. 19 presents that bottom-up ICT architecture is planned to model a city based on a very small scale to a large scale, from building components to neighborhoods. The proposed F2c2C-ICT architecture shows that the bottom Fog-Layer across the city scale has a minimum desirable latency level for the network communications, and their bandwidth and the resource capacities are restricted in sorts of the processing and storage abilities. Going up to the higher layer gives an undesired latency level for the network communications and increases the processing and storage capabilities by multilevel technologies withing upper layers. Finally, the security vulnerabilities are at the high-level at the bottom layer of the city and moving to the upper layer provides more facilities for treating obtained data and data resources.

4.5.1 Data Management Architecture

In this subsection, we intend to model the effective data management architecture for the ZEN center and their related pilots. This data management architecture must provide several facilities to meet the requirements of the ZEN center, as shown below:

- The architecture must cover all conceptual level of ZEN pilots;
- The architecture must support different data types in the ZEN center;
- The architecture must manage all IoT- and non-IoT data sources;
- The architecture must deal with several databases distributed repositories across multiple geographical locations in Norway.

Concerning our D2C-ICT architecture, we understand the hierarchical architecture provides a flexible number of layers according to the city structure or the business model requirements. Accordingly, a recommended ZEN D2C-DM architecture consists of three layers of architecture (namely Fog-Layer, cloudlet-Layer, and Cloud-Layer) based on the F2c2C-ICT architecture and their related data management architecture as depicted in Fig. 19 and described below. Also, several data repositories across the ZEN center pilots are shown in Fig. 20 that those data repositories must organize through our data management architecture.

As we discussed earlier and briefly mentioned here, Fog-Layer is responsible for managing IoT and real-time data sources by using Fog technologies in all pilots' cities. cloudlet-Layer is a middleware platform available for organizing non-IoT and last-recent data sources and combining them with IoT and real-time data locally in the same city as IoT-data sources. Finally, Cloud is a place to organize and access all city pilots' historical data sources and make the possibility to attach them with other non-city data sources concerning the business and data privacy requirements.

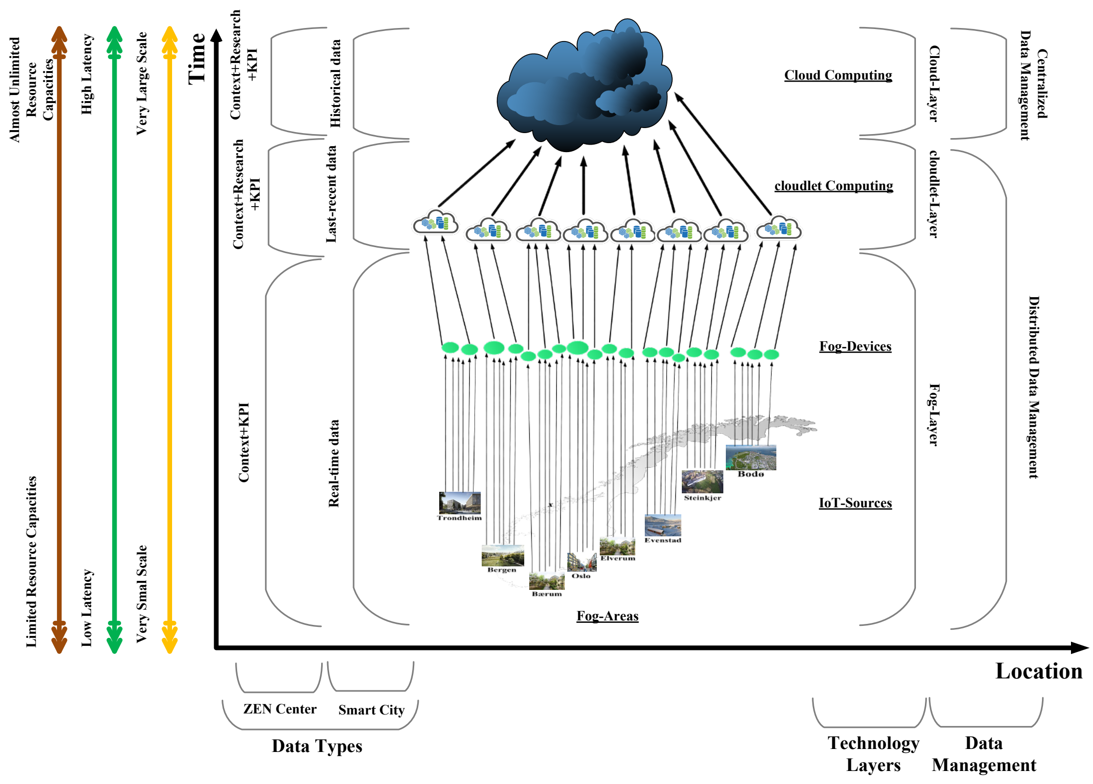


Figure 19 The F2c2C-DM architecture through the smart city scenario [11]

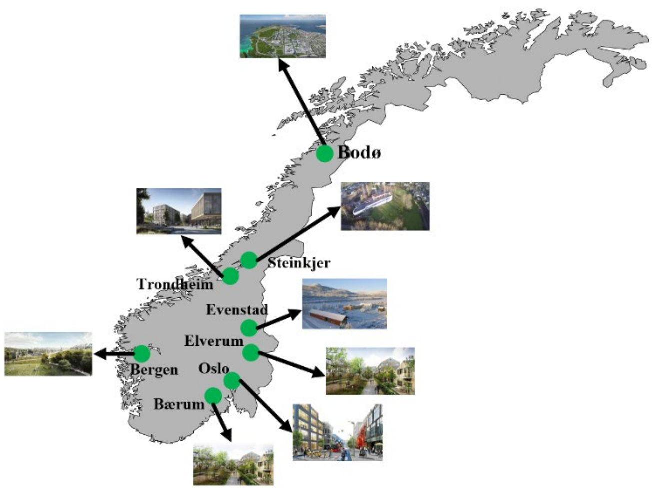


Figure 20 The ZEN pilots city location

4.5.2 Software Services architecture

We also want to provide facilities for various software services that are predicted to be created for the ZEN center and their related pilots in the near future through F2c2C-ICT architecture and related data management architecture, as shown in Fig. 21. The ZEN software services may then be launched through various city-data sources and types in multilevel F2c2C-ICT architecture, as shown in Fig. 21.

As explained earlier, the F2c2C-DM architecture contributed data according to its age and their produced location scope in a city, ranging from real-time to last-recent and historical data. As displayed in Fig. 21, we introduced distributed-to-centralized software services management (D2C-SM) on the right-hand-side of F2c2C-ICT architecture. The D2C-SM architecture followed the D2C-DM architecture idea that showed that all city-data form real-time to last-recent, and historical data could be organized from the pilot's smallest city-scale location at Fog-Layer to the top largest scale of our pilot cities at Cloud-Layer. Our proposed D2C-SM architecture has three main software services layers concerning data privacy and user requirements, as described below.

- "Local Software Services": Developers can work with the locally stored data across the Fog-Layer at the edge of networks to build required software services for each pilot concerning data privacy requirements. Also, cloudlet layers may build various services, including critical and private, in this layer, as shown below.
- "Critical Software Services": Critical services may be lunched inside a city of the pilot at cloudlet- and Fog-Layer. Some examples of critical services are healthcare, fire, and accident. So, the software services may be launched in this layer through the lowest network communication latencies and their bandwidth requirements.
- "Private Software Services": Private and privacy-aware services can be built by utilizing private/ locally stored data in this layer near the city-data sources and citizens. So, we can improve data privacy issues within this sort of private software services for the pilots.
- "Historical Software Services": This layer handles the largest scale of collected city-data at cloud data storage platforms from all city pilots. This layer can also connect to other non-city data sources to build public software services for all relevant customers.
- "Combined Software Services": This layer allows developers to utilize all obtained city-data sources. For example, in [43], the authors described special medical software services that require data from all medical sensors, including real-time, local, and private city-data and Cloud data including historical, large scale, and public city-data. Usually, this kind of software services feeds the history and background of a particular data type in the Cloud platform, and then this city-data can be combined with the local/real-time city-data concerning the required data privacy.

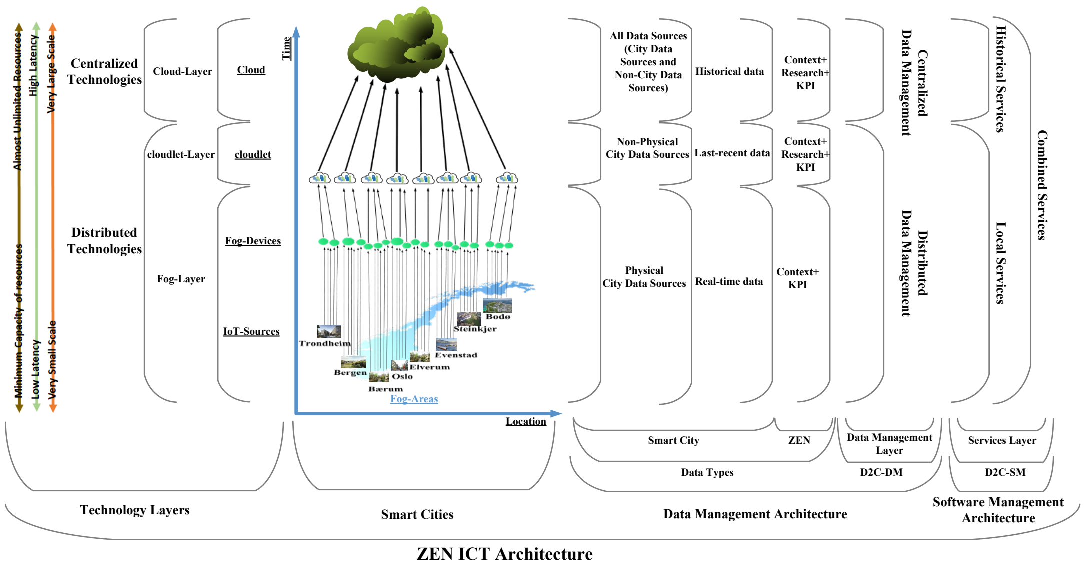


Fig. 21. Building software service through F2c2C-ICT architecture for ZEN [41]

4.5.3 Monitoring system and their related ICT controller

As we addressed in Subsection 2.3.2, the D2C-ICT architecture must be responsible for the management of "data/database," "resources," "network communication," and "software services" in smart cities. As shown in Fig. 22. Our proposed F2c2C-ICT architecture represents an ICT "Integrated and Intelligent Control and Monitoring of IoT (I2CM-IoT)" box at the cloudlet layer to handle all four main management blocks across the multilevel layer of edge-to-cloud orchestration. We decided to position the I2CM-IoT box in the cloudlet-Layer because the cloudlet is in the mid-layer between edge to cloud platform and available in a city. Therefore, we imagine that we can manage all city and cloud resources from city-scale to centralized scale concerning the user and business demands. Finally, the ICT I2CM-IoT box gives data discovery opportunities and data monitoring mechanisms from distributed to centralized schema on the city-scale.

The I2CM-IoT may manage all city requirements through different ICT KPIs, including “network traffic and performance,” “resource performance,” “data age,” “data location,” and “cost calculator.” More details about each ICT-KPIs explained below:

- Network traffic and performance: This category estimates network traffic cost and their related latencies through a set of parameters across multilevel layers of F2c2C-ICT architecture for data discovery purposes in large-scale IoT networks of smart cities. This means the cost shows how fast it is possible to find the relevant city-data sources through different edge-to-cloud layers. The result of network traffic and performance KPIs are relevant to different management blocks of resource, data, and network communication.
- Resource Performance: This category estimates different costs of resource performance usage, e.g., CPU usage, by a set of KPIs across multilevel layers of our F2c2C-ICT architecture. The result will determine the most suitable accessible resource that may help manage resources and software services block.
- Data Age: This category finds relevant city-data sources based on the produced data time, including real-time, last-recent, and historical. As we discussed earlier, each city-data type will produce in different layers of edge to cloud orchestration. The result may be useful for data management block.
- Data Location: This category explores the city-data sources based on the requested data’s location across multilayers of edge to cloud. These actions may be useful for the management of data and resource blocks in smart cities.
- Cost Calculator: This category intends to measure the cost of data within several different parameters such as data access, data price, etc. through different layers of F2c2C-ICT architecture. For instance, some of the produced city-data may not be free and also has special access permission.

The “I2CM-IoT” box proposal may connect with our proposed ICT architecture and ZEN definition guideline and ZEN Tool concerning data monitoring and discovery requirements.

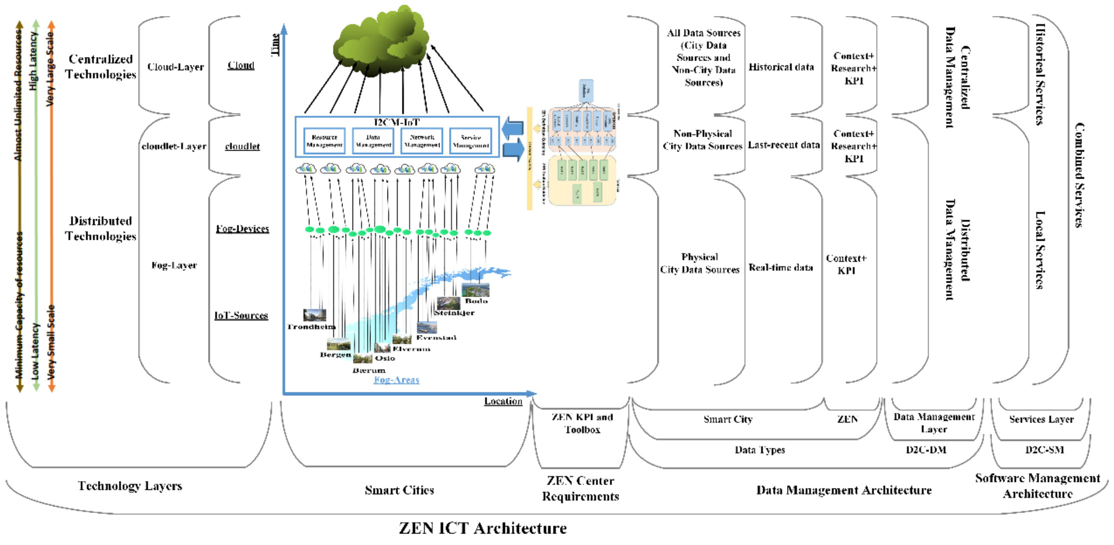


Fig. 22. The proposed monitoring system and related I2CM-IoT Management Block [42]

5 Future Works and Efficiency of our Proposed D2C-ICT Architecture

The efficiency of our proposed F2c2C-ICT architecture for smart cities is varying. It opens several possibilities for future studies and development, including developing software services in smart cities and enhancing the effectiveness of ML and AI techniques through multilevel ICT architecture, as described below. In addition, regarding UN sustainable development goals, the result of this proposal and contribution with our use-case (ZEN centre) may address the requirements of "sustainable cities and communities," "affordable and clean energy," and "industry innovation and infrastructure."

5.1 Developing Software Services for the ZEN

Deciding where and how the software services should be built in a smart city can be divided into four main steps. The different steps can be seen in Fig. 23 and will be discussed in the following part of this section.

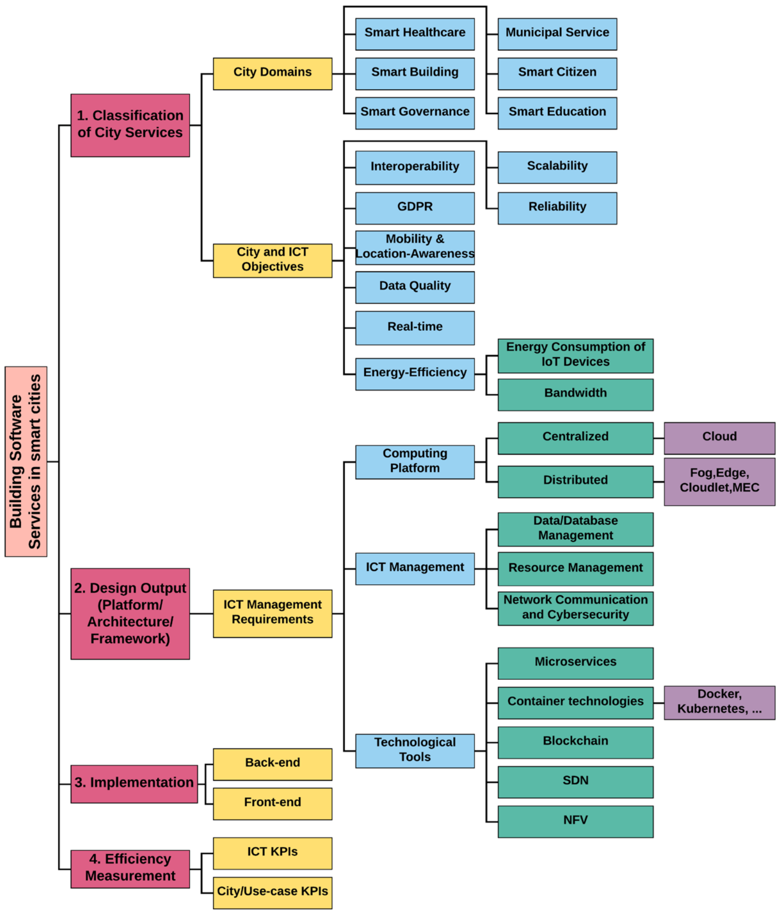


Figure 23 Four main steps for building software services in smart cities

According Fig. 23, the ICT architecture and its data management is in a vital position to develop software services in smart cities and the core research question is that "*how can we build software services in large-scale ICT networks of a smart city from smallest to largest city scale? The solution must provide an integrated solution by using a multilevel, distributed, and centralized technology in combination with a different scope of ICT management strategies.*"

All the complexities, as mentioned above, motivate us to look for the design, implementation, and operation of integral solutions for "large-scale ICT architecture and its data management to develop software services in smart cities through edge-to-cloud orchestration."

More detailed information is available on https://fmezen.no/3scity-e2c-workshop-2020/ and confirmed by several well-known committees and technical program committee members.

5.2 Applying multilevel ML and AI Techniques within and across F2c2C-ICT architecture

The number of ML and AI algorithms and techniques are growing and have been continuously changing and improving smart predictive approaches through different use cases, domains, and scenarios, such as Smart Cities. However, one of the main limiting factors preventing ML tasks is the need for huge and diverse training datasets. In Smart Cities, widely distributed IoT devices networks are continuously capturing various environmental city events and producing many data/datasets. Collecting many datasets may provide several facilities for the Smart Cities data stakeholders, but several challenges arise when the exponential growth of available city-data is stored in the local and centralized data storage media. Due to the computational and memory limitations of IoT devices and low processing abilities at the edge of the Smart City networks, IoT devices are often not capable of running and managing complex ML and AI techniques and algorithms at the edge of the Smart City networks. Finally, sometimes citizens and data stakeholders may not like to share their personal information with others in a public platform (e.g., Cloud technologies).

Recently new calls have been made to design large-scale IoT networks management in the Smart Cities from a very small scale (distributed) to a large scale (centralized) of the city. Edge-to-Cloud computing orchestration may offer a splendid solution to manage Large-Scale IoT networks in Smart Cities, such as data, resources, and software/services, and make it possible to apply complex ML and AL techniques and algorithms at the edge of networks. This solution is also useful to create and train datasets from Edge to Cloud concerning city-data privacy and other related issues.

Edge-to-Cloud orchestration may be put forward with different architectural layers in Smart Cities, such as Fog and cloudlet. There are several opportunities for building predictive model approaches based on distributed-to-centralized learning approaches to train the datasets through different ML and AI techniques through different architectural layers from Edge-to-Cloud orchestration. Therefore, this makes it possible to predict and manage different Smart Cities requirements, such as resource allocation and consumption, and intelligent cyber-attacks predication.

In the case of the combined and hierarchical computing platform and architecture (i.e., Fog-to-Cloud or Fog-to-cloudlet-to-Cloud) from the distributed-to-centralized schema, it is necessary to define the architectural framework for predicting the Smart Cities requirements based on various ML and AI techniques. This framework makes numerous facilities to analyze varieties of datasets from different business domains. Therefore, it is essential to define and design that kind of architectural framework for combined Edge to Cloud technologies systems to manage and predict Smart Cities' requirements, demanding a specific effort from the research community. Currently, two main approaches can be applied through different architectural layers of Edge-to-Cloud orchestration to train varieties of datasets as well as using different ML and AI techniques, as described below:

- *Replicated Learning approach:* Each IoT node/device produces data – which must be sent to the centralized node for training and prediction. Therefore, the centralized node receives and aggregates all data from IoT nodes and builds (a) training model(s), and only that model(s) is(are) shared from the centralized server to all distributed IoT nodes. In summary, replicated learning enables Edge devices to share their local data to the centralized node collaboratively, and the centralized node learns a machine learning model, but keeps all the data on the storage media of the centralized node, instead of storing data on the local distributed nodes. Replicated Learning may come with the following advantages if the effective middleware platform/controller will be designed in between Edge-to-Cloud computing technologies:
    - It may make it possible to build more accurate ML models that are closer to the city-data sources in comparison with building ML models in the Cloud computing platform;

    - It may improve the predication level of resources closer to the city-data sources;
    - It may be possible to store city-data in the middleware platform instead of sending to the Cloud data storage;
    - It improves data privacy efficiency on the middleware platform;
    - This approach optimizes system bandwidth and data availability between geographically distributed data centers in the middleware platform.
- *Federated Learning approach:* Federated learning is a family of ML algorithms that has the core idea: a connected network exists in which there is a central server node. Each of the IoT nodes/devices creates data – that must be used for training and prediction. Each node trains a local model, and only that model is shared with the server, not the data. In summary, Federated learning enables Edge devices to learn an ML model collaboratively but keeping all the data on the IoT devices itself, instead of moving data to the Cloud computing technologies. Federated Learning has the following advantages:
    - Ability to build more accurate models faster;
    - Low latency during inference;
    - Privacy-preserving;
    - Improved energy efficiency of the devices.

More detailed information is available on https://fmezen.no/3scity-e2c-workshop-2021/ and confirmed by several well-known committees and technical program committee members.

6 Implementation

This section demonstrates a way to adapt our proposed ICT architecture and its data management to the ZEN pilots.

This section is divided into two primary subsections. First, we explain the overview of the ZEN pilot's implementation through three different use cases. This highlights how data flow and movement can be done through our D2C-DM architecture. Second, we explain the first simulation of our large-scale ICT architecture.

6.1 Overview of the ZEN pilot's implementation

This subsection is divided into three primary subsections. First, we describe our first use case that is at the micro-level and constitutes several physical data sources (including sensors). The second use case is placed in the meso-level and deals with several physical and non-physical data sources. Last, the location of the last use case is at the macro level.

6.1.1 Use case 1: Micro Level (The Living Lab)

The Living Lab is located at NTNU University in Trondheim, Norway. This pilot provides a test facility for the researchers at the ZEN center. The Living Laboratory is in periods occupied by real persons using the building as their home. The main focus is on the occupants and their use of novel building technologies, including smart control of installations and equipment, the interaction of end-users with the energy systems, and other aspects.

In the first example, we will use the living lab pilot as a case to illustrate the F2c2C-DM architecture adaption. As shown in Fig. 24, the living lab is located at the micro-level (only building level). The living lab pilot data is currently produced by different sensors (IoT-Sources) at the micro-level. Fig. 24 shows the researchers can deal with context data (as real-time data) to build their research data concerning KPI data and their requirements. Finally, for future purposes and usage, the researchers can store the newly obtained data (their research data) in the cloudlet technology and save this newly obtained data in their local storage media.

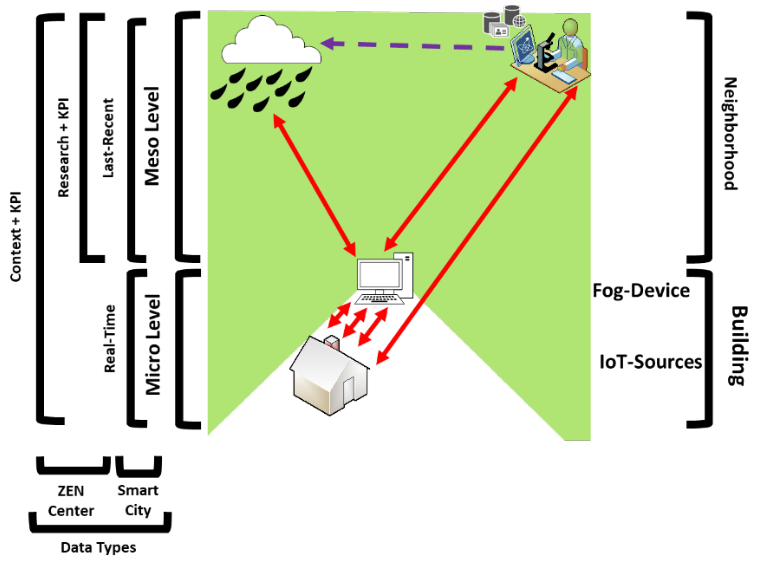


Figure 24 F2c2C-DM architecture through the living lab scenario

6.1.2 Use case 2: Meso Level (Campus Evenstad)

The campus Evenstad pilot is located in a rural area of Stor-Elvdal municipality [1]. The campus consists of seventeen buildings in 61,000 m2 of land. The campus provides a facility for administration, education, and sport, student housing and building operation. This campus aims to optimize energy production for the ZEN center.

In the second example, we will use the campus Evenstad pilot as a case to illustrate the F2c2C-DM architecture adaption. As depicted in Fig. 25, the campus scope is in the meso level (including building and their neighborhoods level). Fig. 25 illustrates that huge amount of data is generated by the different IoT-Sources of seventeen buildings and their related neighborhoods. The researchers can then use the context data (real-time data) to build their research data concerning KPI data as well as store this data in their local storage media and their related cloudlet storages. Note that all data types (including context, research, and KPI) of the Evenstad pilot is available by online access through its interface for different researchers through cloudlet technologies.

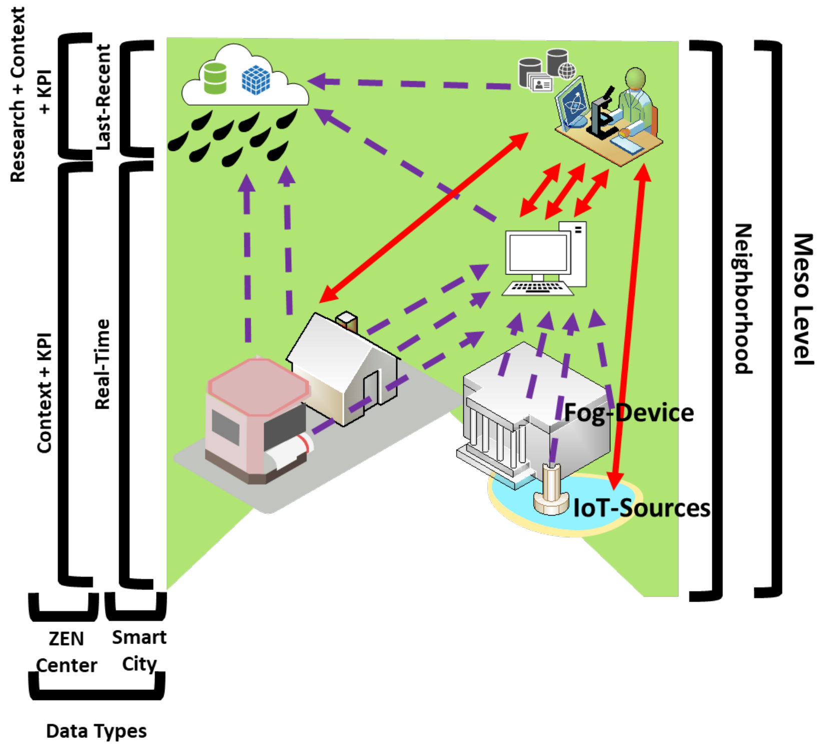


Figure 25 F2c2C-DM architecture through the Campus Evenstad scenario

6.1.3 Use case 3: Macro Level (partners and/or smart city users)

At the macro level, the ZEN partners might request to collect all data from each/all pilots of the ZEN centers by a single request. Therefore, we can use the Cloud technologies' potential to keep all context, research, and KPI data in their repositories for any future process and requirements. Consequently, all partners or other data stakeholders of the ZEN center can access the interface to use all obtained data from the pilots. Also, many software and services can be launched by all obtained data (including historical, real-time, and last-recent data) in this level because the Cloud technology could reach from the top to the bottom of the F2c2C-DM architecture as depicted in Fig. 26.

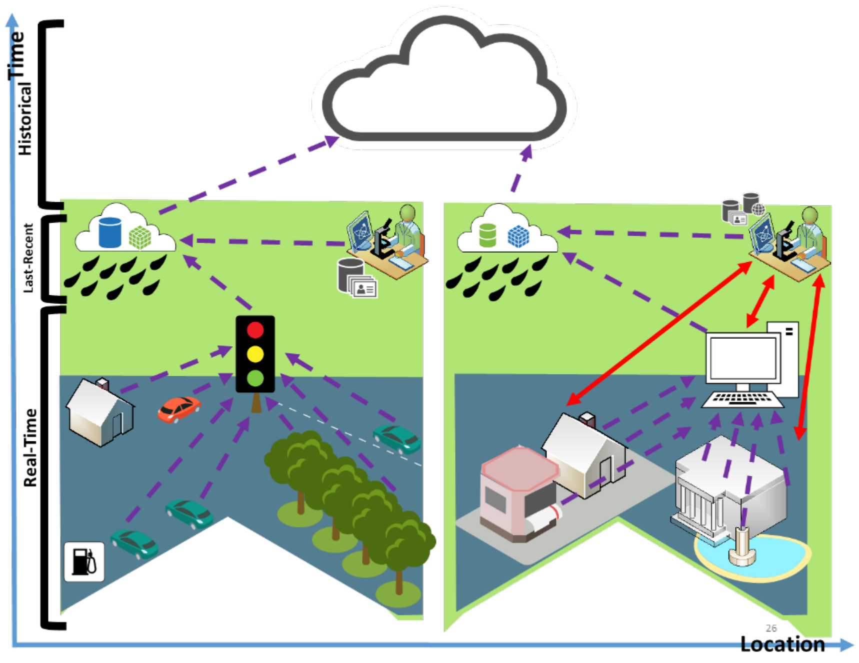


Figure 26 F2c2C-DM architecture through the smart city scenario

6.2 Scenario description for our ICT architecture proposal

In the ZEN scenario, the scenario description is depicted based on Oslo and Trondheim's two city pilots. This scenario comprises a Fog-Layer with edge nodes and sensors that read temperature data. This scenario demonstrated an example of smart city networks with nine nodes, as shown in Fig. 27 and described below:

- Those nodes are one Cloud node, two cloudlet nodes, and six Fog nodes, including two Fog-Leader nodes and four edge nodes.
- Three control units are available in our scenario, one for each of the cloudlet nodes and one for the Cloud node.

We also follow the base of smart cities' networks, which is necessary to be geographically distributed. The nodes' physical location is as following:

- The Cloud node and its controller are positioned in a Digital Ocean droplet located in Frankfurt, Germany.
- The Oslo cloudlet node and its controller are located on a personal computer resided in a building network in Oslo, while the Trondheim node and its controller are run on a raspberry-pi inhabited in a building network in Trondheim.
- Finally, the Fog nodes are installed with the raspberry-pi devices resided in Oslo and Trondheim's same building networks.

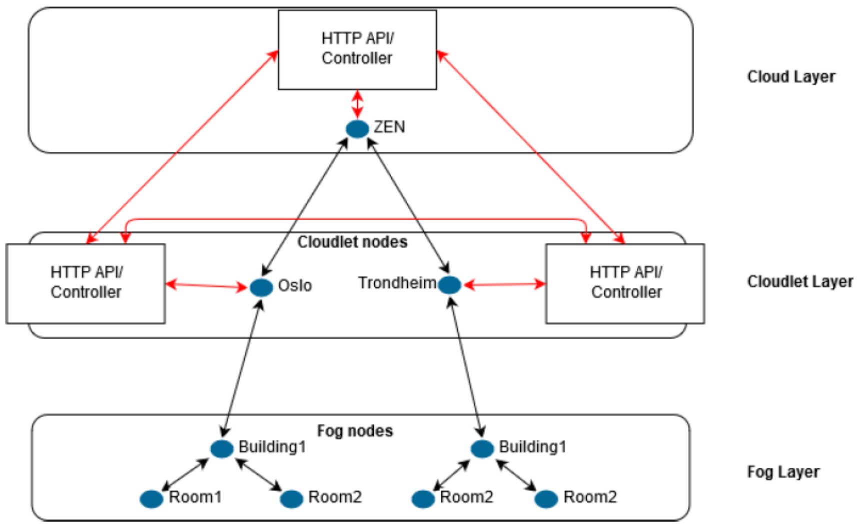


Figure 27 Our scenario description in two different cities in Norway

Two apartments are available in the "building network" of our scenario. Each apartment holds a raspberry-pi device to monitor the apartment temperature, as shown in Fig. 28. Besides, a server room stands in the basement of the building. The platform is intended to manage the sensor devices' temperature data and provide an interface to monitor the platform's data.
The obligations of the platform for our scenario are as regards:

- Create an interface to accumulate temperature data from sensors;
- Build data processing for the obtained data and perform data compression to minute-basis rather than seconds;
- Provide an interface to ask and retrieve the data;
- Create a web application for citizens to deal with the system to illustrate the ICT and data management platform;
- Make an external interface for other platforms to retrieve data for future usage.

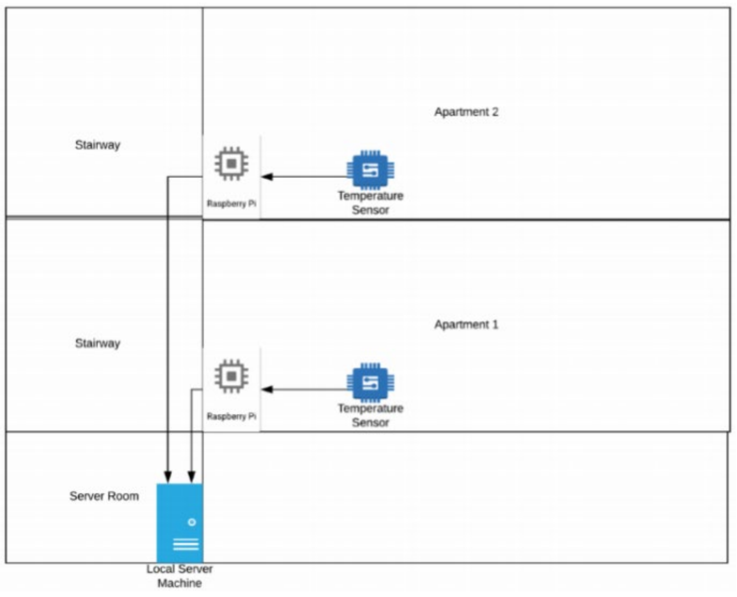


Figure 28 Our scenario for two apartments in a building network

6.2.1 Large-Scale ICT Management Architecture

This subsection aims to show how large-scale ICT management could be done based on the management block available in Section 2.3.2. This section is organized into four main subsections, as described below.

Data/Database Management

As shown in Fig. 29, the elements inside the swarm cluster section are created to be containerized and scaled within replication procedures to befit the system's workload demands. It also gives the API to facilitate receivable and retrieval of data. The request handler can also retrieve the appropriate data, such as real-time, last-recent, or historical data.

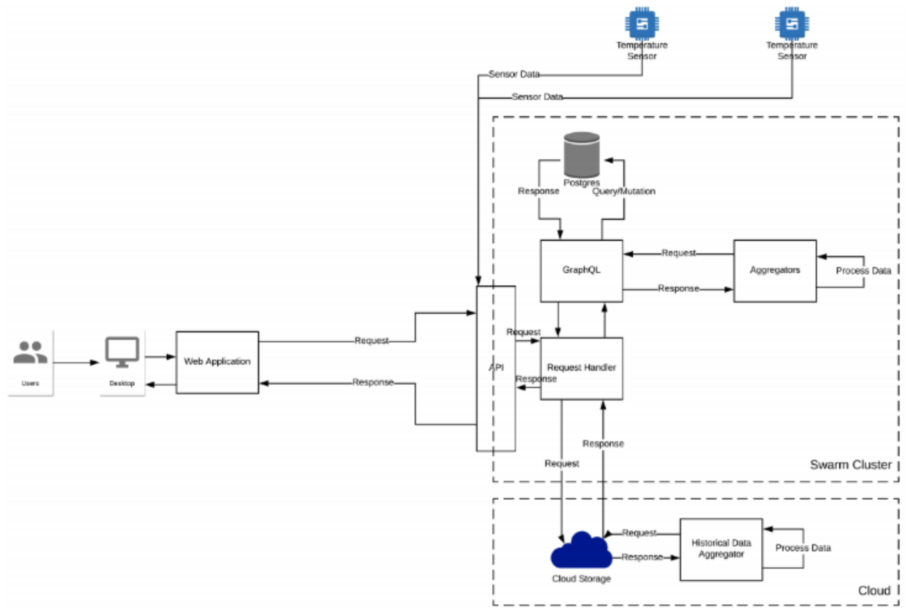


Figure 29 Proposed scenario for data/database management

The below points show how data aggregation, data storage, and data discovery as some primary data/database management block data can be made in our scenario.

- Multilevel data aggregation and data storage: Data are produced with diverse temperature sensors in our scenario depicted in Fig. 30. The obtained sensors data is stored nearby the data sources in open-source relational database management in Fog-Area, which is the Postgres platform through multiple tasks from "GraphQL," "request handler," and related "API." Besides, the "Aggregator" components are interconnected to the "GraphQL." Accordingly, the platform may apply data aggregation before saving the sensors data in the Postgres database. The "request handler" is also attached to the "Cloud Storage" at Cloud technologies. "Cloud Storage" is accountable for the permanent storage facilities of historical data. Consequently, we can save and update the collected data in Fog and Cloud technologies within the different update mechanisms and "request handler" methods. "historical data aggregator" components are also attached to the "Cloud Storage" platform as the second data aggregation component.

- Data discovery: we used the MAUT cost model [43]. As shown in Fig. 30, the I2CM-IoT box worked as a controller for our scenario and can be queried and respond from our ICT architecture's multilevel layers. The I2CM-IoT has three primary components, "request handler," "cost model," and "routing." In Fig. 31, the internal architecture and how the various components cooperate is depicted, where solid arrows point the internal control flow of the I2CM-IoT box and request and response to the I2CM-IoT stippled arrows determine the queries that the routing component might perform.

We realized by our implementation that if city-data is available in local storage in the city, it is better to find and send data to the citizens from the edge of networks instead of taking from the cloud storage platform. It means connecting to the internet for either getting data/information from the cloud storage or transferring data back to the cloud platform is costly, and much delay appears through connecting and receiving data from cloud storage, which is somewhere outside of the city.

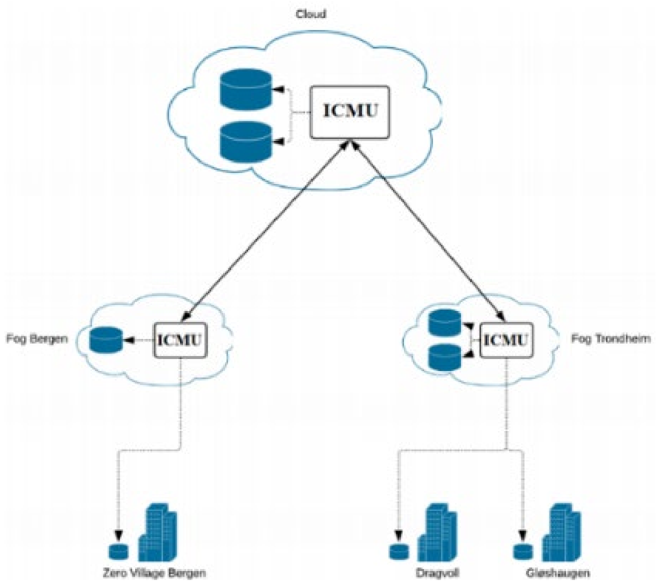


Figure 30 Our scenario for data discovery

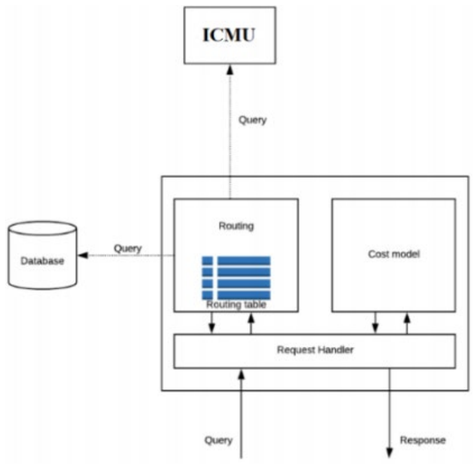


Figure 31 Internal architecture of an I2CM-IoT

Resource Management

The MAUT cost model, as we discussed above, is also beneficial for the resource management block.

The cost model pseudo-code algorithm is mentioned in Fig. 32. So, it says all the alternatives data will be found through edge-to-cloud orchestration. The citizens can define a set of strict constraints in the filter settings for each data discovery. If the alternative cuts a constraint, it is removed, and nothing is attached to the last results. If the data is not filtered out, it repeats through the ranking criteria, and the score will be calculated through the weight and utility function of each ranking criteria, totaling it to the final score. The final result will show the best match of data for the citizens.

By adopting the MAUT [26] ranking approach, we estimate the participating system resources' matching score in our scenario. Concerning the greatest scores, the system allocates the most relevant resources for performing the task(s) and achieving some job(s) and provide the demanded service(s) among the system subscribers. This all cost model ranking shows the benefits of this study for the resource management block as well as was for the data management block.

ALGORITHM 1: The cost model ranking calculation

```
$filters$: a list of strict constraints;
$critera$: a list of criteria to rank with;
$alternatives$: a list of alternatives;
$result$ = [];
for a in alternatives do
    add = true;
    for f in filters do
        if a breaks constraint f then
            add = false;
        end
    end
    if add == true then
        score = 0;
        for c in criteria do
            score = score + (c ranking for a)
        end
        result.append({score: score, data: a})
    end
end
```

Figure 32 Proposed algorithm for the cost model ranking calculation

Network Communication Management

This subsection demonstrates our implementation related to the network communication management block into three main subsections, including naming, routing, and controllers.

- Naming: The proposed content naming structure in the network supports hierarchy and is based on a URI scheme with the "/" character delimiting distinct parts of the hierarchy. The naming base is the content names of the sensor devices at which they were created. The global naming scheme is as following:

/zen/<city>/<building>/<room>/<sensor>/<timestamp>
Example: /zen/oslo/building1/room1/temperature

It is necessary to remark that the timestamp is a unique address in the addressing schema, so the content names also are globally unique. This unique addressing system is required for the D2C-ICT architecture because city-data may exist in multiple layers. As the produced city-data gets older in terms of data production, it will ultimately be pushed and saved in the cloud storage at Cloud-Layer. Consequently, since the network will have to discover content names uniquely at the Cloud-Layer, local names are prevented in this system.

- Routing: The basic interest routing in the network is tree-structured because of its hierarchical structure. Below are some examples of how addressing mechanisms can work and route in multilevel layers of edge-to-cloud orchestration to find the best data for citizens.
    - An interest hosted at the Cloud-Layer will be sent down the layers until the data exists in its most basic form.
    - Though it is achievable to request data from the controllers of nodes lower down the network's edge close to the data sources, the cloudlet nodes can also send interest upwards to the Cloud if data should not exist at the lower layers. The lower a node is in the network's edge layers, the more accurate it is routing to that node. This indicates that requests are pushed out to the network edges. For example, a cloudlet node will retrieve data from the below layers before moving to the upper layer. In this case, if the content is not in its local storage and is in its related domain.

- The nodes execute routing based on the longest prefix matching in their Forwarding Information Base (FIB). e.g., if a request for data comes to the cloudlet ICN relay node in Oslo, it will first look at its local storage before pairing the request name with the longest match in its FIB. If the content name is in its domain, this indicates that first, it will always send the interest to the Fog node below, only moving to the Cloud if the content does not exist. Also, because the Cloud's prefix is only "/zen," this will be the best pair for content names outside the cloudlet domain, expecting that they will only be sent to the Cloud, skipping the sub-network altogether. Table. 2 shows The FIB of the cloudlet node in Oslo.

Table 2 FIB for the cloudlet node in Oslo

FIB for the Oslo cloudlet	
Prefix	**Interface**
/zen	cloud node
/zen/oslo/ building1	fog node

While request data has existed in a node, the network jumps to the reverse path to give the citizens the required data.

The network routing works functional and straightforward, but it does not perform any form of responsive routing. That is where the cloudlet controllers appear in our scenario. The cloudlet estimates the possible paths to data and chooses the one with the cheapest latency. It is essential to remark that this policy is only working, while data has several paths. For example, data in the related domain as the cloudlet controller does not have various data paths and is not included by the policy. If there is a call for data outside the domain, at least last-recent data, the network's cloudlet, or Fog-Layer data founds. Without the latency policy assumption, the only route to retrieve the data would be to move through the Cloud. With the policy assumption, it is understandable to move directly to the data's cloudlet controller that presented the lowest latency path. An example of this scenario can be shown in Fig 33. that the red arrow depicts the new forwarding rule appended to the FIB.

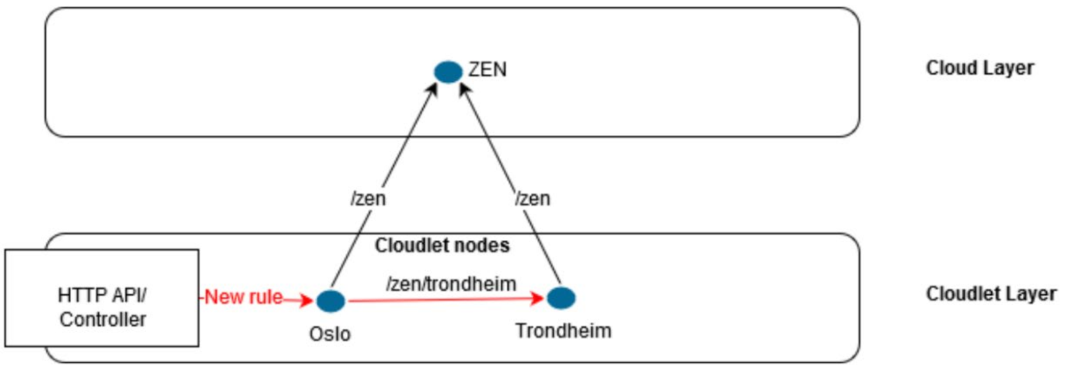


Figure 33 New forwarding rule added to the Oslo ICN relay node

To sum up, our middleware platform's efficiency can be highlighted by using the cloudlet controllers for responsive routing. First, it might lower the request latency, understanding as the path moves straight from Oslo to Trondheim for data rather than into the Cloud in Germany. Second, it decreases total network traffic by avoiding the Cloud, decreasing bottlenecks' risk at the Cloud level.

- Controllers: The controllers in the network perform three functions:
  - First, the cloudlet controllers perform as software-defined networking (SDN) inspired routing controllers for their corresponding ICN relay nodes. While a request defines the lowest latency

policy, it is their responsibility to choose the most suitable path and subsequently update the FIB of their ICN relay node.
  - Second, they perform as API endpoints for data requests, expecting that data consumers can request data at either of the three controllers.
  - Third, they build valid content names based on deterministic data, i.e., the data consumer's timestamp intervals.

The controllers are scripts executed in Python and coded as simple HTTP servers that listen to a port. They can, via PiCN, communicate with their ICN node to give interest packets or renew the FIB of the node. Also, they can ping the other controllers in the network to renew their latency. Furthermore, they have simple lists that keep the summary of their respective sub-networks and Cloud and cloudlet locations. ////The controllers in the network implement three features: First, the cloudlet controllers' function as SDN-inspired routing controllers for their respective ICN relay nodes. When a request specifies the lowest latency policy, it is their responsibility to select the best path and subsequently update the FIB of their ICN relay node. Second, they function as API endpoints for data requests, meaning that data consumers can request data at either of the three controllers. Third, they create valid content names based on deterministic data, i.e., the timestamp intervals provided by the data consumer.

Service/Software Management

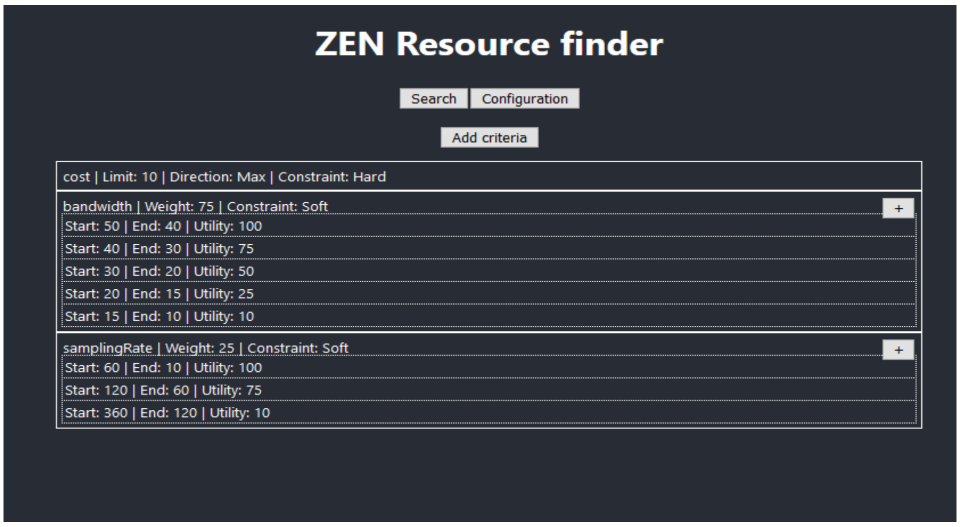
D2C-ICT architecture offers the facility to build services in multilevel layers of F2c2C-ICT architecture. Additionally, we can share and monitor available data in each layer through the web-based application. As shown in Fig. 28, we create a web application. This web application presents a user interface for the citizens to find and monitor their related data from edge-to-cloud orchestration.

6.2.2 User Interface and results

This subsection presents the results of implementing the F2c2C-ICT architecture prototype in giving a part of the solution, and some related testing is done, as demonstrated below.

Data/database Management

- ZEN cost model user interface: The user interface might be through the API provided by the I2CM-IoT, which takes the form of a JSON data in an HTTP POST request with the search query and the cost model configuration. The design consists of two pages, an explanation of how it works, as shown below.
  - *"Configuration" page:* The cost model configuration page is where the user can create the inputs of different criteria for the cost model, as shown in Fig. 34. The "Add Criteria" button allows adding either a "soft criteria," which is a criterion that is used as part of the MAUT portion of the cost model, with weights and utility functions used for ranking, or a "hard constraint" that acts as a filter for the solution. In the figure, there is a significant constraint on the cost that specifies a max cost of 10 and two soft criteria on bandwidth and sampling rate, with their utility intervals below them, listed under the buttons. The "+" button is used to add utility intervals to the soft criteria. This configuration is used when searching for data in a search page.



cost | Limit: 10 | Direction: Max | Constraint: Hard

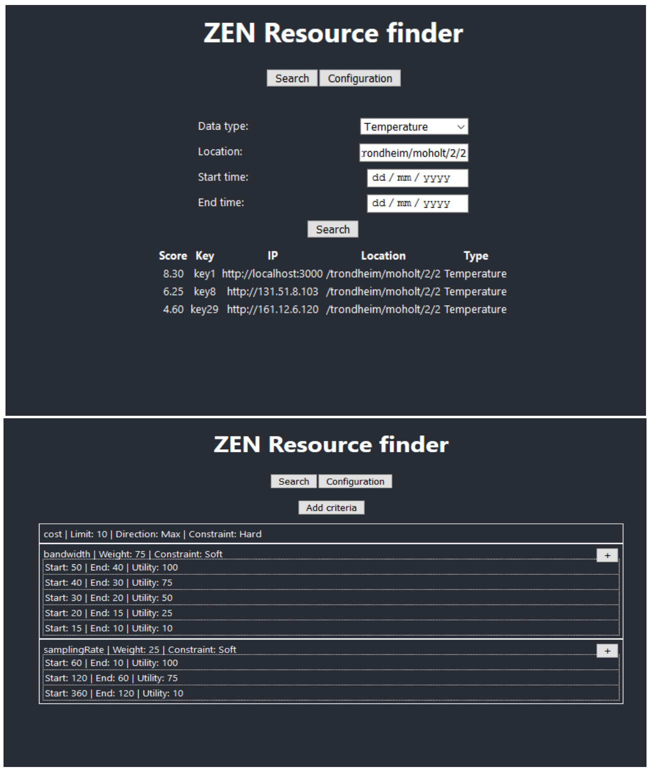
bandwidth | Weight: 75 | Constraint: Soft
Start: 50 | End: 40 | Utility: 100
Start: 40 | End: 30 | Utility: 75
Start: 30 | End: 20 | Utility: 50
Start: 20 | End: 15 | Utility: 25
Start: 15 | End: 10 | Utility: 10

samplingRate | Weight: 25 | Constraint: Soft
Start: 60 | End: 10 | Utility: 100
Start: 120 | End: 60 | Utility: 75
Start: 360 | End: 120 | Utility: 10

Figure 34 “Configuration” page for the cost model implementation

- *“Search” page:* As shown in Fig. 35, the search pages specify search queries in terms of “data type,” “data location,” and “data age.” At the same time, the cost model configuration uses different KPIs to filter and rank alternatives.

Below it is the search input for the data search, the search parameters are the data type, the location that data was generated, and the time period in which it was generated. Below the search, “the result of our search" is listed. Also, the "score of each result" concerning the MAUT approach is calculated.



Data type: Temperature
Location: :rondheim/moholt/2/2
Start time: dd / mm / yyyy
End time: dd / mm / yyyy

```
Score  Key    IP                      Location                  Type
8.30   key1   http://localhost:3000  /trondheim/moholt/2/2  Temperature
6.25   key8   http://131.51.8.103    /trondheim/moholt/2/2  Temperature
4.60   key29  http://161.12.6.120    /trondheim/moholt/2/2  Temperature
```

cost | Limit: 10 | Direction: Max | Constraint: Hard

bandwidth | Weight: 75 | Constraint: Soft
Start: 50 | End: 40 | Utility: 100
Start: 40 | End: 30 | Utility: 75
Start: 30 | End: 20 | Utility: 50
Start: 20 | End: 15 | Utility: 25
Start: 15 | End: 10 | Utility: 10

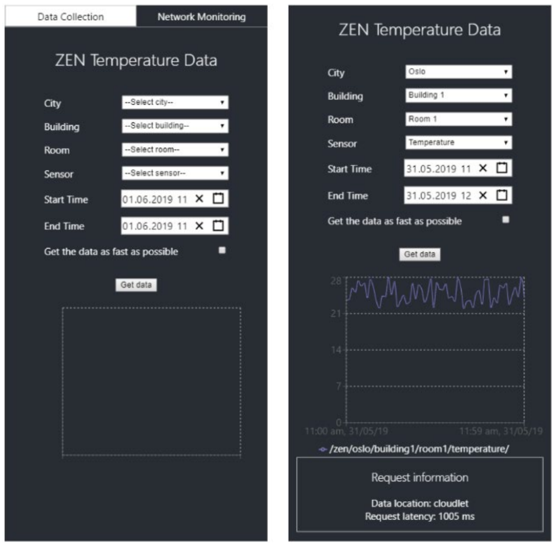
samplingRate | Weight: 25 | Constraint: Soft
Start: 60 | End: 10 | Utility: 100
Start: 120 | End: 60 | Utility: 75
Start: 360 | End: 120 | Utility: 10

Figure 35 “Search” Page for the cost model Implementation

As the results illustrate, for particular preferences and needs, data stored in the distributed part of the system might be better than the one stored in the Cloud. Also, using the cost model implementation can help decide when using data from the Cloud is best, and when using data from the distributed repositories is the best alternative.

Network Management

- ZEN network discovery: The design pages consist of two main pages, as described below.
  - *"Data Collection" page:* As shown in Fig. 36, there are several drop-down menus that the user can click to specify what data is to be requested, from which city, building, room, and sensor. This includes a date and time picker, enabling the user to request a time interval for the data. This is restricted to the time and date of the dataset. The start time is inclusive, and the end time is exclusive. Next, there is a checkbox marked "get the data as fast as possible," and when it is checked, the controller is instructed to use the latency policy when fetching data. This only applies to data outside of the domain, the controller that is either last-recent or earlier (real-time). Otherwise, it is only found in the Cloud. Therefore, checking the checkbox instructs the controller to compare the latency to the Cloud and cloudlet and choose the lowest latency path. Leaving it unchecked means that requests for data outside the domain always goes through the Cloud.
  - Next, the "get data"-button performs the HTTP request to the controller. When returned, the data is visualized in a chart, with the x-axis representing the time and the y-axis representing the value of the data (in this case in Celsius). The legend below the chart shows the content name of the request. Finally, some information about the request is provided, so where the data was requested is in the network layer and the latency of the request.



ZEN Temperature Data

City --Select city--
Building --Select building--
Room --Select room--
Sensor --Select sensor--
Start Time 01.06.2019 11
End Time 01.06.2019 11
Get the data as fast as possible

ZEN Temperature Data

City Oslo
Building Building 1
Room Room 1
Sensor Temperature
Start Time 31.05.2019 11
End Time 31.05.2019 12
Get the data as fast as possible

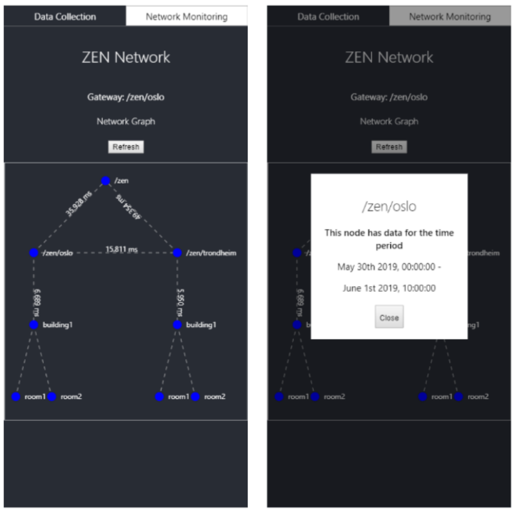
/zen/oslo/building1/room1/temperature/

Request information

Data location: cloudlet
Request latency: 1005 ms

Figure 36 "Data Collection" page for the ZEN network discovery

  - "Network Monitoring" page: The network monitoring view provides a tree-structured graph view of the entire network. It can be seen to the left in Fig. 37. The "refresh" button causes all the links and nodes to be updated. All the links (excluding the edge links, seeing as the nodes there are in the same device) are provided with latencies to monitor each link's general latencies. When clicking on a node in the graph, the node's name, and the time interval of the data that it contains is revealed, as seen to the right in Fig. 37. Note that this information is only available when clicking on the nodes in the cloudlet and Cloud-Layers, as only they have controllers that contain the information on the time interval of data stored in the nodes.



/zen/oslo

This node has data for the time period

May 30th 2019, 00:00:00 -

June 1st 2019, 10:00:00

Figure 37 "Network Monitoring" page for the ZEN network discovery

6.2.3 Results and discussions

The purpose of this chapter is to focus on the network management block to evaluate and discuss the practical usage efficiency advantages of the distributed-to-centralized naming and routing mechanism model prototype, in particular our proposed middleware platform. Therefore, the results of the measurements administered on usage efficiency will first be described. Secondly, the D2C-ICT architectural approach will be presented with the measurements as a base, estimating the architecture's advantages and drawbacks compared to a centralized ICT architecture. Finally, a discussion of the decisions created while developing the architecture, concerning architectural choices and prototyping and technology options, will be discussed.

Results

The results for the four measurements will be shown here below.

Response Time and Request Latency for Real-Time Data

The first measurements were arranged on requests for real-time data from the cloudlet controller's sub-domain, indicating that no routing outside the cloudlet's sub-network was applied. The request applied in these measurements converts to an interest of the form */zen/trondheim/building1/room1/2019-06-01T00*, collected at both the cloud and the Trondheim cloudlet controllers. This determines that the requested controllers were the cloud and Trondheim cloudlet controllers, while the data was hosted in the edge node called room1 under the Trondheim cloudlet controller, as shown in Fig. 38.

The two controllers' response time depicted in Fig. 38 shows quite clearly that there is a small, but notable difference in response time at the cloud and cloudlet levels. This highlights that any more significant difference in request latency is likely to, in large part, be caused by the location of the data consumer. The result shows that the response time is lower at the cloudlet than the cloud layer.

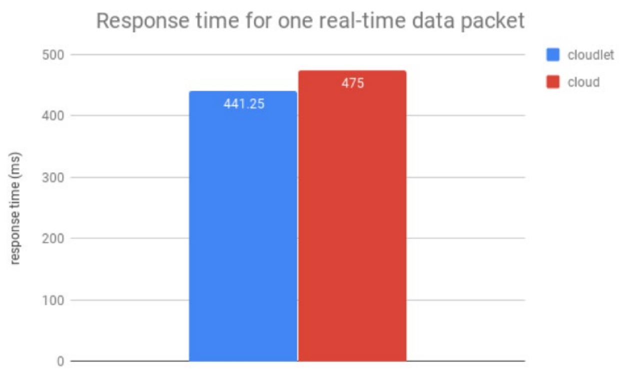
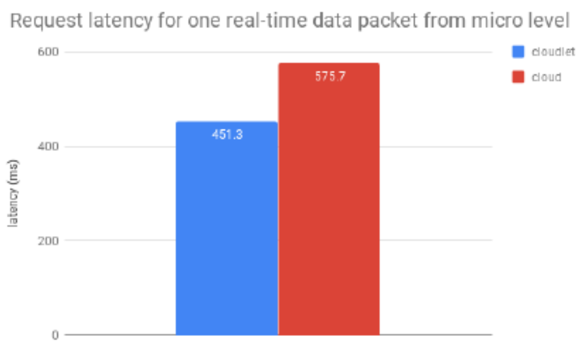


Figure 38 Average response time for one real-time data packet

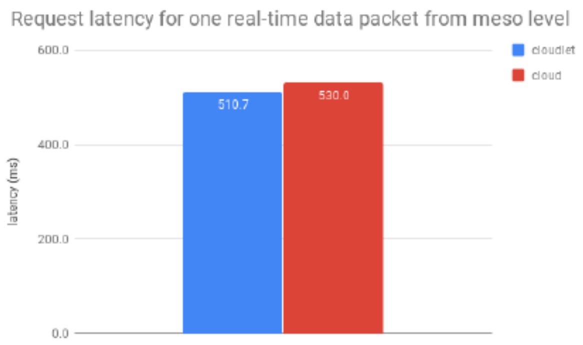
As shown the average latency of the requests from the three different conceptual levels in Fig. 39., there is significantly lower latency for requests to the cloudlet than the cloud while the data consumer is at the micro-level, close to the network's edge. The lower latency is almost similar to the difference in ping from the micro-level (times two), indicating that the data consumer location and not the system response time is the most defining variable for request latency. Next, in the figure, the data consumer is at a meso-level. The request latencies are now much closer, but there is still a slight benefit to requesting data from the cloudlet layer controller. Repeatedly, the difference in latencies is approximately corresponding to the difference in ping times two, from the meso-level. Lastly, the data consumer is at a macro level. Here the request latencies have been reversed, and there is somewhat lower latency for requests to the cloud layer controller. This again corresponds to the difference in ping from the macro-level times two. A summary of the combined latencies for the different layers' requests can be seen in Table. 3.

Table 3 Latency in milliseconds for requests for one real-time data packet to cloud and cloudlet layer controllers

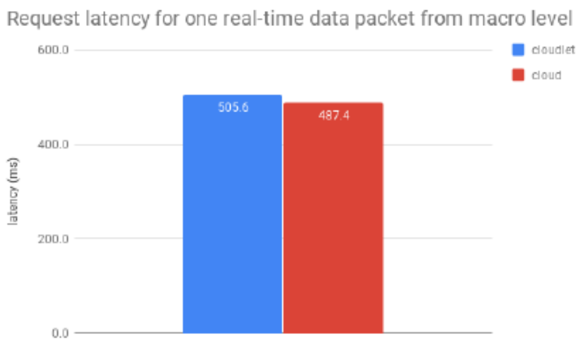
Entry Point	Micro Level	Meso Level	Macro Level
Cloud	575.7	530.0	464.7
Cloudlet	451.3	510.7	505.6



(a) Micro level



(b) Meso level



(c) Macro level

Figure 39 Latency for one real-time data packet from three different levels

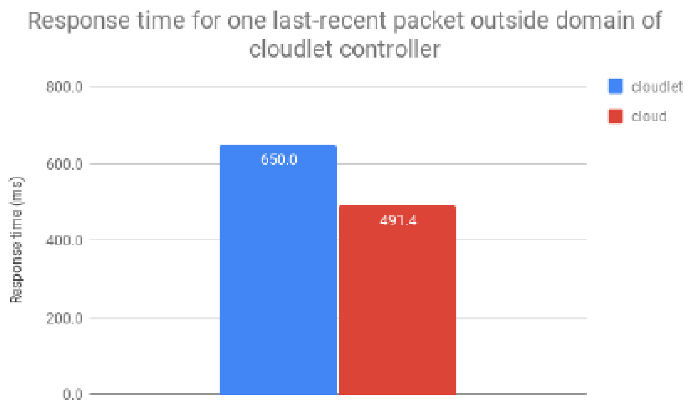
These measurements present the benefit of holding cloudlet controllers to assist real-time data to citizens at the micro- and meso- levels. Apart from the apparent benefit of network traffic of not going through the cloud layer, there are lower latencies for requests from both the micro- and meso- levels. Also, there is a somewhat lower request latency at the macro level while going through the cloud layer, indicating that these requests are best served through the cloud concerning the latency perceived by the citizens. Concerning overall network traffic, the cloudlet is the best option in all three cases as it decreases the number of internal network links utilized.

To summarize this section, the cloudlet controllers can give better latencies for citizens near the network edges than the cloud, while also offloading the cloud connection by going the requests lower in the network. However, when it happens to citizens at the macro level, these might be best served by a cloud connection, depending on the user's needs or system. A service that connected the data consumer with the most suitable controller based on their geographical position could guarantee that the data consumer got the data from the controller that was optimal concerning latency. Combined, distributing requests from the micro- and meso- levels while serving macro requests centralized through the cloud could provide a better user experience for citizens at all three levels that were tested, as well as make for a less congested network, distributing the traffic more evenly compared to a centralized cloud approach.

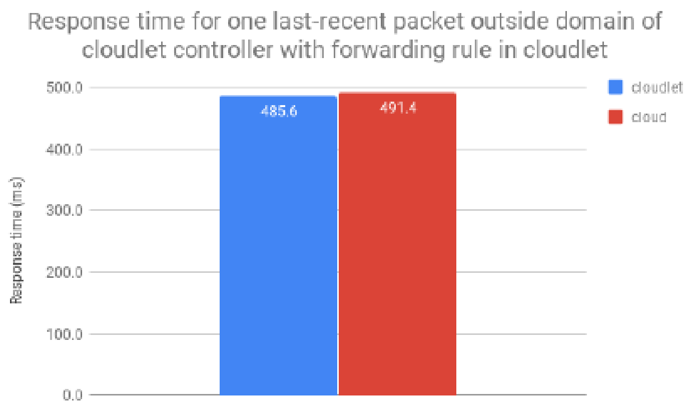
Response Time and Request Latency for Last-Recent Data

This measurement's objective was to evaluate the routing efficiency of a request received at the cloud and cloudlet layers while the interest is for last-recent data outside the controller's domain. The request applied in these measurements converts to an ICN interest packet of the form */zen/oslo/building1/room1/2019-05-31-00*, received at both the cloud and the Trondheim cloudlet controllers. The interest, therefore, is for last-recent data that is located in the Oslo cloudlet. This measurement will serve to compare being able to route interest packets from one cloudlet to another with a forwarding rule, going laterally rather than through the cloud, with having to go through the cloud to get to other cloudlets, meaning the use of routing policy by the controllers will be evaluated. As with the previous measurement, the system's response time will first be presented, after which the latency from the micro-, meso- and macro-level will be measured.

The system's response time with and without a forwarding rule can be seen in Fig. 40. The figure shows a considerable delay when requesting data at the cloudlet controller without a forwarding rule to the other cloudlet compared to the cloud, while when the forwarding rule is present, there is no significant difference between the two. This can be seen when looking at Fig. 40, where it is evident that without being able to forward laterally, the cloudlet will have to go through the cloud layer to get to the other cloudlet and back, making it a much longer trip. The cloudlet would have to do two internal network hops to the data (to the cloud, and to the other cloudlet) and two hops back, for a total of four hops without a forwarding rule. In comparison, with a forwarding rule, it is only necessary with two hops.



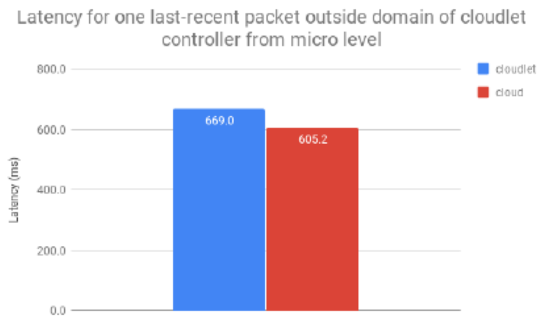
(a) Without forwarding rule



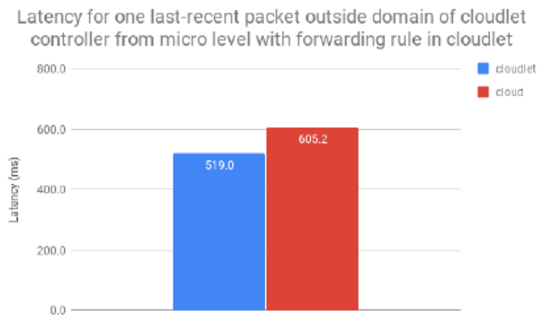
(b) With forwarding rule

Figure 40 Response times for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule

As for latency, beginning with the data consumer at the micro-level, Fig. 41 clearly shows the benefit of inter-cloudlet routing. Here the difference is a whole 150 milliseconds on average between the cloudlet with and without a forwarding rule, meaning that it is much more efficient to route the interest from one cloudlet to another directly rather than going through the cloud. Note that while the first graph shows that the cloudlet is less efficient than the cloud in retrieving the data, the second shows the reverse. Here, the cloudlet is more efficient than the cloud in retrieving the data for a micro-level data consumer. This would indicate that the cloudlet would best serve a data consumer at the micro-level.
Fig. 42 shows the latencies when the data consumer is at the meso level, with and without a forwarding rule in the cloudlet. Again, we see how much less efficient it is to request data at the cloudlet layer when there is no forwarding rule present, compared to otherwise. The difference is around the same as with the first measurement at the micro level, about 170 milliseconds. The figure also notes that when the forwarding rule is present at the cloudlet, it can be just as efficient to request the cloudlet as the cloud, suggesting that a data consumer in the meso level can be just as well served by requesting the cloudlet as the cloud for data in another city than the cloudlet.

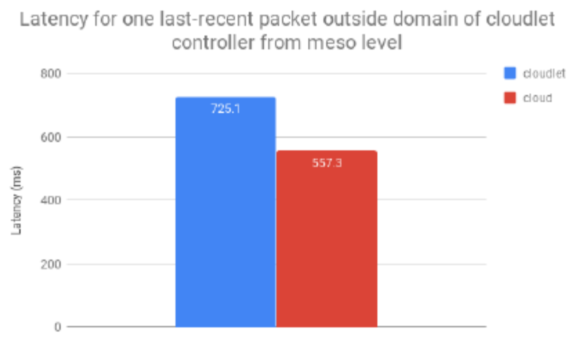


(a) Without forwarding rule

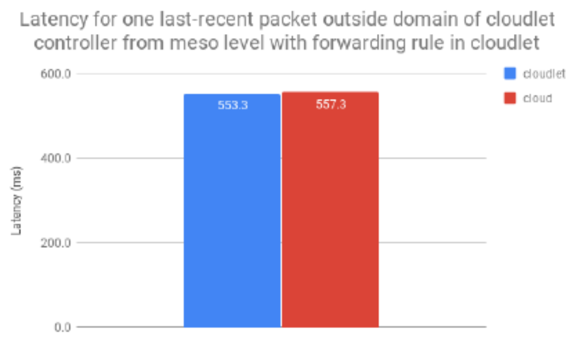


(b) With forwarding rule

Figure 41 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the micro level



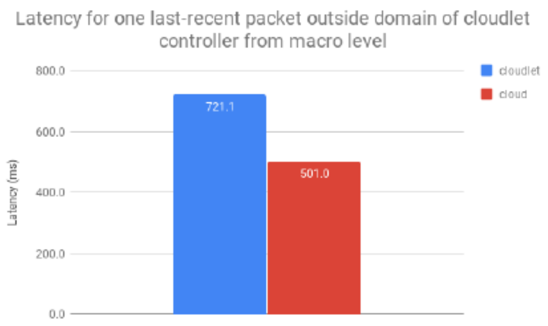
(a) Without forwarding rule



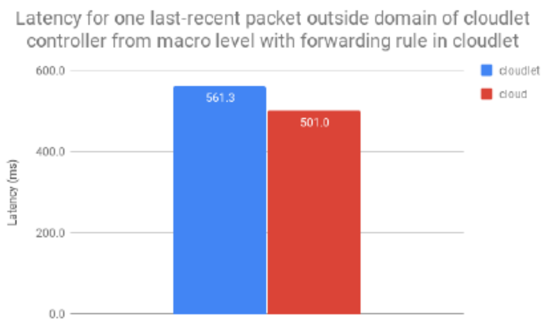
(b) With forwarding rule

Figure 42 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the meso level

The last measurements in Fig. 43 present the latencies while the data consumer is at the macro level, with and without a forwarding rule in the cloudlet node. The difference between the two cloudlet measurements is about the same as the earlier measurements, around 160 milliseconds. The figures show how much the difference in inefficient data delivery can be reduced between the cloudlet and the cloud. However, even with the forwarding rule, data delivery is more than 10 % slower from the cloudlet than the cloud to a data consumer at the macro level. This suggests that the cloud best serves a data consumer at this level.



(a) Without forwarding rule



(b) With forwarding rule

Figure 43 Request latency for one last-recent data packet outside the domain of the cloudlet controller with and without forwarding rule. From the macro level

A summary of the latencies at all levels can be seen in Table 4. The table clearly shows the difference in latency for the cloudlet with and without a forwarding rule.

Table 4 Latency in milliseconds for requests for one last-recent data packet to cloud and cloudlet layer controllers

Entry Point	Micro Level	Meso Level	Macro Level
Cloud	605.2	557.3	501.0
Cloudlet	669.0	725.1	721.0
Cloudlet with rule	519.0	553.3	561.3

To sum up, as was the case with the measurements for real-time data in the last section, the significant benefit of a cloudlet controller that can take data requests can be seen for data consumers at the micro and meso levels. However, this is only when the cloudlet can forward the request directly to the other cloudlet, proving the necessity of being able to perform routing at the cloudlet layer for data requests outside the domain of the cloudlet. Otherwise, the latencies will be much higher for all levels of data consumers at the cloudlet than at the cloud. At the macro level, the table again shows that the benefit of cloudlet requests is negated, suggesting that a different policy should be applied for those data consumers than the others.
The impact on network traffic is essential to note. Without forwarding to the other cloudlet, requesting data from the cloudlet is not only less efficient in terms of latency, but it also creates more network traffic than requesting data from the cloud. This is due to the request going through the cloud and down to the other cloudlet, meaning the interest makes two hops rather than the cloud's one hop. On the other hand, requesting data at the cloudlet when the cloudlet is forwarding directly to the other cloudlet creates an equal amount of network traffic as at the cloud, but most importantly: it can offload some requests from the cloud to the cloudlet while being in some cases more efficient for data delivery. Even when it

is less efficient, as is the case with a data consumer at the macro level, this benefit is worth considering. The potential for reducing bottlenecks at the cloud can be a significant advantage.

Response Time and Request Latency for Historical Data

The following measurement is for historical data, that is to say, data that exists solely in the cloud layer. This serves to evaluate the efficiency of such requests received at the cloudlet layer and forwarded to the cloud versus requests that are received directly at the cloud layer. The request used in these measurements converts to an interest of the form */zen/trondheim/building1/room1/2019-05-01-00*, received at both the cloud and the Trondheim cloudlet controllers. This indicates that the data was located in the cloud, making it historical data. Once again, the requests were conducted at a micro, meso- and macro- level.

As the response times of the cloud and cloudlet shown in Fig. 44, it can be determined that there is a significantly lower response time for requests for historical data at the cloud level, not surprisingly because the data is local to the cloud node and can therefore be returned immediately without going into the sub-network.

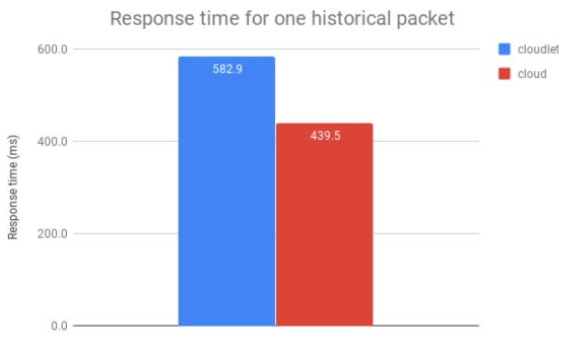


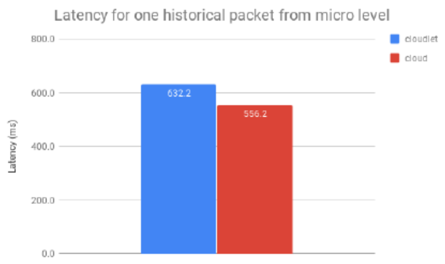
Figure 44 Response time for one historical data packet

However, as shown in Fig. 44, the latency difference is somewhat lower than the difference in response time when the request is coming from a data consumer at the micro-level. This is to be expected due to the more considerable distance of the data consumer to the cloud. The latency is, however, still lower when requesting the cloud.

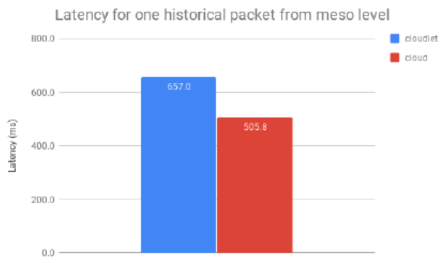
As might be intuited, the difference in latency between the cloudlet and cloud-only grows when the data consumer is at a meso and macro level, as shown in Fig. 44. This is in line with the ping times shown in table 5 in [44] and suggest that data consumers requesting historical data should go to the cloud layer directly to minimize the request latency. Due to this fact, and the fact that requests going through the cloudlet layer use more of the network links, it can be suggested that requests for historical data are best served directly from the cloud, no matter where the data consumer is, in order to reduce both latency and network traffic.

Lastly, a summary of the combined latencies for the different layers' requests can be seen in Table 5. This highlights what has been seen in all the measurements so far that the further the data consumer is from the cloudlet, the lower the cloudlet's ability to serve data becomes efficient. Conversely, the closer the data consumer is to the cloudlet, the lower the cloud's ability to efficiently serve data becomes. The difference here in comparison with the other measurements conducted is that at no point does the cloudlet outperform the cloud.

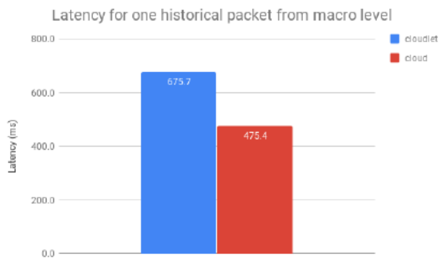
Interestingly, the cloudlet's response time can be seen to be much higher than that of the cloud in Fig. 45. Intuitively, these times should be closer, since the two requests are in some sense the reverse of each other: the last-recent data request at the cloud node goes to the cloudlet node and back for data retrieval, while the historical data request at the cloudlet node goes to the cloud node and back. This should, in essence, the same trip, but in reverse. However, due to how the FIB of the cloudlet node is structured, it will first forward the interest to its sub-network (because the sub-network node has the longest matching prefix of the interest) after receiving a no-content response will it forward it to the cloud layer. This introduces a significant delay, as well as unnecessary network traffic.



(a) Micro level



(b) Meso level



(c) Macro level

Figure 45 Latency for one historical data packet from three different levels

Table 5 Latency in milliseconds for requests for one historical data packet to cloud and cloudlet layer controllers

Entry Point	Micro Level	Meso Level	Macro Level
Cloud	556.2	505.8	475.4
Cloudlet	632.2	657.0	675.7

Response Time for Larger Quantities of Data

Up until now, the measurements have only been conducted on requests for one data packet. However, it is useful to know how the network handles requests for larger quantities of data as well, as this is one of the main features of the timestamp converting that can be done in the controllers of the system. In this case, the only thing measured was the system's response time because it is the author's opinion that this serves to sufficiently illustrate the efficiency of the system in response to larger data requests. The response times are a magnitude larger than the ping times from the different data consumer layers (seconds rather than hundreds of milliseconds), meaning that the predominant part of total request latency would be the response time of the system.

The measurements were conducted for three different sizes of data: 60 packets, 90 packets, and 120 packets. These then totaled to 95 kB, 142 kB, and 189 kB, respectively. The data requested was in all cases the most recent data, meaning that in the first case, the data was 1 real-time and 59 last-recent packets, in the second case 1 real-time, 59 last-recent and 30 historical packets, and in the third 1 real-time, 59 last-recent and 60 historical packets. All requests were for data in the domain of the Trondheim cloudlet, meaning they corresponded to the content name partial */zen/trondheim/building1/room1/temperature*. The cloudlet requested was the Trondheim cloudlet. Fig. 46 shows the measurements. What is evident from the figure is that while the cloud exhibits a fairly linear growth in response time as the data size grows, the cloudlet has a much steeper response time growth. This would suggest that for the cloud, the most time-consuming part of the requests was fetching data from the lower layers, that is, the cloudlet and the fog. Vice versa for the cloudlet becomes much more expensive in terms of time when more data is in the cloud.

Interestingly, for 90 packets, twice as many packets from the cloudlet and lower as from the cloud, the response time appears to be about the same. It can be concluded that a data consumer is best served from the cloudlet if there is less than about half as much data to be fetched in the cloud than in the cloudlet and lower. If more data is to be fetched in the cloud than that, it becomes prohibitively expensive for the cloudlet to serve it.

However, as was the case with the historical data measurements, the structure of the cloudlet FIB might play a role in the large latency increase for the cloudlet when more data must be fetched from the cloud. The cloudlet continues to its sub-network with the requests before moving to the cloud. Another limitation of the system, in conjunction with this, makes the measurements problematic: There is no form of chunking for larger data requests. This means that the controller creates an interest packet for each data packet, which is then treated the same by the ICN node each time. This results in that every packet that is not local to the node must be routed individually, introducing a large potential overhead. While it might be correctly pointed out that even if this is a limitation, it is the same for both the cloud and the cloudlet, the limitation of the cloudlet FIB would mean that as there are more historical data interest, the difference in latencies between the cloud and cloudlet will only grow. If the system could perform data chunking, a single request would suffice, for instance, 30 historical data packets, potentially making the routing time and consequently response time for such requests much lower.

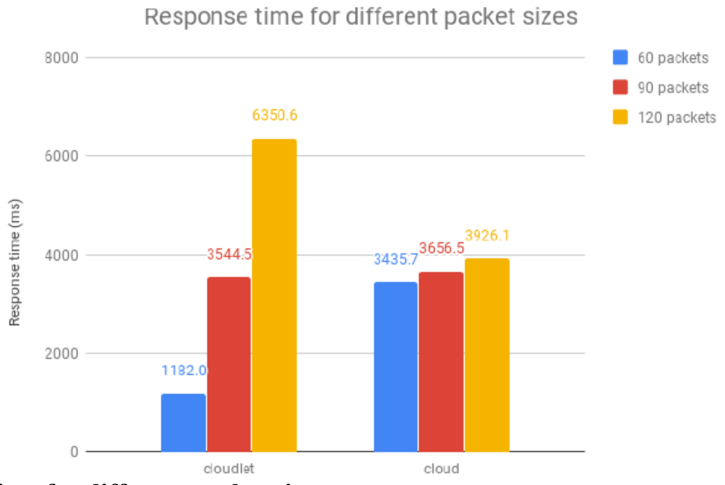


Figure 46 Response time for different packet sizes

These limitations aside, it is our opinion that the measurements for larger data requests can be used to shed light on some of the potential benefits and limitations of the system. Larger data requests for data at the fog and cloudlet layer are more efficiently delivered from the cloudlet, and even when some of the data is in the cloud, it can be equally efficient to request the cloudlet. Also, utilizing the cloudlet for such data requests will ensure that as many interests as possible are kept at a lower level in the network, reducing traffic to the cloud. On the other hand, while the limitations of this prototype make any discussion of the delivery efficiency of larger quantities of historical data difficult, it will at some point be worth considering if a request that involves large amounts of historical data is not best served from the cloud, both in terms of potential network traffic and efficiency of delivery. In terms of device constraint, that may exist at the cloudlet layer in a real-world implementation.

Discussions

In the measurements conducted, the architecture's usage efficiency was assessed in terms of data access in multilevel D2C-ICT architecture. The experiment was conducted based on system response time and user-perceived latency to assess the efficiency of data access and delivery. Other benefits of the architecture can be mentioned too, which do not pertain to data access efficiency.

At the most general aspect of the architecture, data storage at lower levels in the D2C-ICT architecture, the cloudlet and fog layer, enables more efficient data access for data consumers in smart cities. This is the enabling factor for all other data access benefits because if data were only stored in the cloud, no more efficient data access than cloud access would be possible anyway. The results also show that enabling data access through the cloudlets makes it more efficient to retrieve data for data consumers in the network's lower layers. For a single real-time data packet in the cloudlet domain, the Table 3 shows that data access is more efficient for data consumers in both the micro (same city) and meso (same country) levels of the network. This can be extended to last-recent data as well, as it stands to reason that data local to the cloudlet can be delivered as fast or even faster than data in its sub-network. Due to the controller-enabled routing deployed in the network, the same can be said for data access for real-time and last-recent data in another cloudlet than the one accessed. Table 3 shows that, when the forwarding rule to the other cloudlet is deployed, data consumers at the micro and meso level are as good or better served by the cloudlet than the cloud.

On the other hand, the results show that for all types of data, from real-time to historical, a data consumer at the macro level will be best served by the cloud. Also, and quite self-evidently, because historical data is local to the cloud, all data consumers' levels will be best served by the cloud for data access to historical data in terms of latency. This is of less importance because historical data tends to be of less time-critical. It is worth evaluating how a policy for dealing with the data consumers' scenarios at different geographical distances to the network should be put in place. For instance, if the data consumer is outside the network, perhaps in another continent, the policy might always serve the data through the cloud.

Lastly, the results show how the system's response time grows when more data is being requested. It shows that when the data requested is mostly real-time or last-recent, data access at the cloudlet can benefit from lower response times. However, as the table shows, at some point when the data is more historical, the response time of the cloudlet will become prohibitively large. Thus, the centralized part of the architecture will be of more importance to these types of data requests, proving how D2C-ICT architecture can combine the benefits of both distributed and centralized architectural approaches. Nonetheless, as in data consumers, it must be considered how a policy for requesting large data quantities should be implemented, and when it is prudent to move the request to the cloud altogether.

The network architecture benefits manifest in other ways, among them in less network traffic. When data can be accessed closer to the data consumers, there is less load on the network because the requests do not have to go all the way to the cloud to get data. However, this is true not only for the results of data requests, as presented in the previous subsection but will also be valid for data storage. Since the data is distributed in multilevel layers, there is no immediate transfer of data from the edges to the cloud when it is created. However, the lack of such data management structures in the prototype cannot be evaluated accurately, only suggested.

Distributing data request reduces bottleneck in the cloud as indicated earlier. Implementing a policy that ensures that data consumers request data from the most appropriate level would serve to make the network more efficient and responsive and distribute the requests throughout the network. This would serve to spread the network usage out more, reducing network load. One of the most limiting factors of centralized architectures would then be much less of a problem: bottlenecks at the cloud. Because requests for data access can be satisfied at the lower layer of the architecture, even in the data in another cloudlet than the one requested, they need not touch the cloud layer, leaving the network links at the cloud layer to handle fewer requests.

Moving on to more general benefits that could not be gleaned by the previous subsection results, we see that the cloudlets make it easy to define data policies at a per-layer or even per-cloudlet basis. The prototype utilizes only one such policy (that of lowest latency), but it opens the door for many more. One could imagine data policies for many different scenarios being imposed at the cloudlet, be it to decrease cost, reduce network load, or maximize data quality. This can be an excellent boon for network administrators and operators when configuring the network, making the network flexible and efficient.

7 Conclusion

This report aims to show the main benefits of designing D2C-ICT architecture using multilevel technologies from edge to cloud in smart cities. To achieve our objective, we followed the below steps:

- First, we went through ICT reference architectures, as described in Section 2. We found that we are interested in focusing on the technical view (vertical landscape) and management blocks (horizontal landscape).
- Second, we realized that the current proposed ICT reference architecture is not suitable for a distributed to centralized schema. The base of this architecture is designed for the centralized cloud technologies platform. Therefore, it is necessary to design a comprehensive ICT reference architecture based on edge to cloud orchestration.
- Third, as the main part of this study, we already proposed several novel ideas about the technical view (vertical landscape) and management blocks (horizontal landscape), including data/database, resource, network communication, and software services management.
- Fourth, we showed how data could be managed and monitored through our proposed D2C-ICT architecture in smart cities' large-scale IoT networks.
- Fifth, we mentioned how we could use the idea of D2C-ICT architecture to ZEN requirements. We highlighted the importance of this study for the future of smart cities and ZEN studies.
- Sixth, we showed some of our implementation of the idea of D2C-ICT architecture in large-scale IoT networks of smart cities. The results highlighted the importance and efficiency of our D2C-ICT architecture for future smart cities in integrated solutions for data/database, resource, network communication, and software services management.
- Seventh, some results about the network management block and efficiency of our middleware platform at the cloudlet layer are discussed in Section 6.
- Finally, there are several ideas for future studies in this area, as mentioned in Section 5.

Acknowledgment

This report has been written within the Research Centre on Zero Emission Neighbourhoods in Smart Cities (FME ZEN). The authors gratefully acknowledge the support from the Research Council of Norway, the municipalities of Oslo, Bergen, Trondheim, Bodø, Elverum and Steinkjer, Sør-Trøndelag county, Norwegian Directorate for Public Construction and Property Management, Norwegian Water Resources and Energy Directorate, Norwegian Building Authority, ByBo, Elverum Tomteselskap, TOBB, Snøhetta, ÅF Engineering AS, Asplan Viak, Multiconsult, Sweco, Civitas, FutureBuilt, Hunton, Moelven, Norcem, Skanska, GK, Caverion, Nord-Trøndelag Elektrisitetsverk - Energi, Numascale, Smart Grid Services Cluster, Statkraft Varme, Energy Norway and Norsk Fjernvarme.

References

[1] E. W. Johnston and D. L. Hansen, "Design lessons for smart governance infrastructures," *Transforming American governance: Rebooting the public square,* pp. 197-212, 2011.

[2] I. FG-SCC, "Setting the framework for an ICT architecture of a smart sustainable city," *Focus Group Technical Specifications,* p. 49, 2015.

[3] C. Perks and T. Beveridge, *Guide to enterprise IT architecture*. Springer Science & Business Media, 2007.

[4] J. McGovern, S. W. Ambler, M. E. Stevens, J. Linn, E. K. Jo, and V. Sharan, *A practical guide to enterprise architecture*. Prentice Hall Professional, 2004.

[5] P. g. Schenkl. (2017). *ICT architecture*. Available: http://www.ict-expert.com/services/ictarchitectureen.php#:~:text=1%20%2D%20ICT%20Architecture,of%20a%20company's%20ICT%20systems

[6] T. Z. A. Abou El Seoud, "Towards Sustainability: Smart Cities In The Egyptian Environment, How Much Smart To Be Smart?," *Journal of Urban Research,* vol. 31, no. 1, pp. 123-142, 2019.

[7] I. Schieferdecker, N. Tcholtchev, P. Lämmel, R. Scholz, and E. Lapi, "Towards an Open Data Based ICT Reference Architecture for Smart Cities," in *2017 Conference for E-Democracy and Open Government (CeDEM)*, 2017, pp. 184-193: IEEE.

[8] E. Johnston, "Governance infrastructures in 2020," *Public Administration Review,* vol. 70, pp. s122-s128, 2010.

[9] J. Jin, J. Gubbi, S. Marusic, and M. Palaniswami, "An information framework for creating a smart city through internet of things," *IEEE Internet of Things journal,* vol. 1, no. 2, pp. 112-121, 2014.

[10] J. Gubbi, R. Buyya, S. Marusic, and M. J. F. g. c. s. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems on elsevier journal,* vol. 29, no. 7, pp. 1645-1660, 2013.

[11] A. Sinaeepourfard, J. Krogstie, S. Abbas Petersen, and D. Ahlers, "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City," presented at the IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, in press.

[12] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Fog-to-Cloud (F2C) Data Management for Smart Cities," in *Future Technologies Conference (FTC)*, 2017.

[13] A. Sinaeepourfard, J. Krogstie, and S. Sengupta. (2020, in press) Distributed-to-Centralized Data Management: A New Sense of Large-Scale ICT Technology Management of Smart City Networks. *Internet of Things Magazine (IoTM)*.

[14] I. T. Union, "ICT Role for Smart Sustainable Cities: Smart Sustainable Cities Training Programme Module 1 SSC-1," Available: https://www.itu.int/en/ITU-D/Regional-Presence/AsiaPacific/Documents/Module%201%20Smart%20Sustainable%20Cities%20Overview%20Revised_SK.pdf.

[15] J. Robberechts, T. Goethals, B. Volckaert, and S. Abbas Petersen, "Developing Software Services in Smart Cities based on Edge to Cloud Orchestration," Master degree, Faculty of Computer Science, Norwegian University of Science and Technology (NTNU), 2020.

[16] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Fog-to-Cloud (F2C) Data Management for Smart Cities," presented at the Future Technologies Conference (FTC), 2017.

[17] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "A Novel Architecture for Efficient Fog to Cloud Data Management in Smart Cities," in *IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 2622-2623: IEEE.

[18] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Data preservation through Fog-to-Cloud (F2C) data management in smart cities," in *IEEE 2nd International Conference on Fog and Edge Computing (ICFEC)*, 2018, pp. 1-9: IEEE.

[19] T. K. Soltvedt, A. Sinaeepourfard, and D. Ahlers, "A Cost Model for Data Discovery in Large-Scale IoT Networks of Smart Cities," in *2020 21st IEEE International Conference on Mobile Data Management (MDM)*, 2020, pp. 348-353: IEEE.

[20] H. Hu, Y. Wen, T.-S. Chua, and X. Li, "Toward scalable systems for big data analytics: A technology tutorial," *IEEE access,* vol. 2, pp. 652-687, 2014.

[21] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, E. Marin-Tordera, X. Yin, and C. Wang, "A data lifeCycle model for smart cities," in *International Conference onInformation and Communication Technology Convergence (ICTC)*, 2016, pp. 400-405: IEEE.

[22] F. Almeida and C. Calistru, "The main challenges and issues of big data management," *International Journal of Research Studies in Computing,* vol. 2, no. 1, pp. 11-20, 2013.

[23] Y. Demchenko, Z. Zhao, P. Grosso, A. Wibisono, and C. De Laat, "Addressing big data challenges for scientific data infrastructure," in *IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012, pp. 614-617: IEEE.

[24] R. Grunzke *et al.*, "Managing complexity in distributed Data Life Cycles enhancing scientific discovery," in *IEEE 11th International Conference on E-Science (e-Science)*, 2015, pp. 371-380: IEEE.

[25] A. Sinaeepourfard, J. Garcia, and X. Masip-Bruin, "Hierarchical distributed fog-to-cloud data management in smart cities," Doctoral thesis, Departament d'Arquitectura de Computadors, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, 2017.

[26] M. M. Rathore, A. Ahmad, A. Paul, and S. Rho, "Urban planning and building smart cities based on the Internet of Things using Big Data analytics," *Computer Networks,* vol. 101, pp. 63-80, 2016/06/04/ 2016.

[27] A. Sinaeepourfard, J. Krogstie, S. A. Petersen, and A. Gustavsen, "A Zero Emission Neighbourhoods Data Management Architecture for Smart City Scenarios: Discussions toward 6Vs challenges," presented at the International Conference on Information and Communication Technology Convergence (ICTC), 2018.

[28] T. V. N. Rao, A. Khan, M. Maschendra, and M. K. Kumar, "A Paradigm Shift from Cloud to Fog Computing," *International Journal of Science, Engineering and Computer Technology,* vol. 5, no. 11, p. 385, 2015.

[29] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, 2012, pp. 13-16.

[30] K. Bilal, O. Khalid, A. Erbad, and S. U. Khan, "Potentials, trends, and prospects in edge technologies: Fog, cloudlet, mobile edge, and micro data centers," *Computer Networks,* vol. 130, pp. 94-120, 2018.

[31] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *The Fifth ASE International Conference on Big Data*, 2015, p. 28: ACM.

[32] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," presented at the Proceedings of the first edition of the MCC workshop on Mobile cloud computing, 2012.

[33] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/Fog Computing Resource Management and Pricing for Blockchain Networks," *IEEE Communications Magazine,* 2018.

[34] A. Sinaeepourfard, J. Krogstie, and S. A. Petersen, "A Big Data Management Architecture for Smart Cities based on Fog-to-Cloud Data Management Architecture," in *Proceedings of the 4th Norwegian Big Data Symposium (NOBIDS)*, Trondheim, Norway, 2018.

[35] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marin-Tordera, "Data Preservation through Fog-to-Cloud (F2C) Data Management in Smart Cities," presented at the Fog and Edge Computing (ICFEC), 2018 IEEE 2nd International Conference on, 2018.

[36] A. Sinaeepourfard, J. Garcia, X. Masip-Bruin, and E. Marín-Torder, "Towards a comprehensive data lifecycle model for big data environments," presented at the Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies, 2016.

[37] *The Research Centre on Zero Emission Neighbourhoods in Smart Cities – ZEN.* Available: http://smart-cities-centre.org/wp-content/uploads/7.2-Arild-Gustavsen-NTNU-The-Reserach-Centre-on-Zero-Emission-Neighbourhoods-in-Smart-Cities.pdf

[38] J. Barceló, J. Casas, D. García, and J. Perarnau, "Methodological notes on combining macro, meso and micro models for transportation analysis," in *Workshop on Modeling and Simulation*, 2005.

[39] M. Lapp. (2013). *Levels of detail.* Available: https://www.memecon.com/levels-of-detail.html

[40] 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," 2019, in press, Available: https://fmezen.no/category/publications/reports/.

[41] A. Sinaeepourfard, S. Abbas Petersen, and D. Ahlers, "D2C-SM: Designing a Distributed-to-Centralized Software Management Architecture for Smart Cities," presented at the Proceedings of the 18th IFIP Conference on e-Business, e-Services, and e-Society (I3E 2019), Trondheim, Norway, 2019, in press.

[42] 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)*, 2019, pp. 760-765: IEEE.

[43] D. Von Winterfeldt and G. W. Fischer, "Multi-attribute utility theory: models and assessment procedures," in *Utility, probability, and human decision making*: Springer, 1975, pp. 47-85.

A Appendix

A.1 Contribution of this proposal with ZEN centre up to December 2020

The report includes and is based on several contributions with ZEN center and its partner as listed below:

- Work on the definitions and KPIs of ZEN as reported in D1.1.1;
- Work on operationalizing the KPIs for the pilots in WP6;
- Initial discussions with WP4 on energy data;
- Initial discussions with the ICT industry partners;
- Contributions to mapping out existing tools of ZEN and their applicability in ZEN, as well as understanding data dependencies and data flows in T1.3;
- ZEN Workshop on tools, planned, used, and developed in the ZEN centre, December 2017;
- Previous work on smart city data management, data quality, and enterprise architecture;
- Several discussions with different research groups in the ZEN center;
- Our new scientific publications (Section 6);
- Presentation of "Pecha Kucha ", FME-ZEN: Seminar of prof. Kjærgaard (Energy informatics), 26 February 2019, Trondheim, Norway;
- Presentation of "Smart Data and Services Presentation", Smart City Day Seminar, 10 December 2018, Trondheim, Norway;
- Conference Presentation of "A Big Data Management Architecture for Smart Cities based on Fog-to-Cloud Data Management Architecture", NOBIDS 2018, 14 November 2018, Trondheim, Norway (https://www.ntnu.edu/nobids) ;
- Poster Presentation of "F2C-DM Architecture", ZEN partnerseminar 2018, 25 October 2018, Trondheim, Norway;
- Meeting with Seemi Lintorp", Trondheim eiendom (Trondheim Kommune), Norway, 14 November 2018, Trondheim, Norway;
- Conference Presentation of "A Zero Emission Neighborhood Data Management Architecture for Smart City Scenario: Discussions Toward 6Vs Challenge", IEEE ICTC 2018, 19 October 2018, South of Korea, (http://2018.ictc.org/);
- Presentation of "Hierarchical Distributed Fog-to-Cloud Data Management in Smart Cities", Gemini IoT Center, 20 Jun 2018, Trondheim, Norway (https://www.ntnu.edu/web/1279812099/phd-seminar-2018).
- Presentation of "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 (IINTEC 2019), Hammamet, Tunisia, December 2019.
- Presentation of "D2C-SM: Designing a Distributed-to-Centralized Software Management Architecture for Smart Cities", 18th IFIP Conference on e-Business, e-Services and e-Society (I3E 2019), Trondheim, Norway, September 2019.
- Presentation of "D2C-DM: Distributed-to-Centralized Data Management for Smart Cities based on two ongoing case studies", Intelligent Systems Conference (IntelliSys) 2019, London, UK, September 2019.
- Presentation of "IoT and Octopus: A new sense of control into smart city solutions", Seminar on Gemini Center on Internet of Things, Oslo, Norway, May 2019.
- Presentation of "Distributed-to-Centralized Data Management through Data LifeCycle Models for Zero Emission Neighborhoods", TopHPC (High-Performance Computing), Tehran, Iran, April 2019.
- Presentation of "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City", IEEE 5th World Forum on Internet of Things (IEEE WF-IoT 2019), Limerick, Ireland, April 2019.
- Presentation of "Smart Data and Smart Services for Smart Cities", "Smart City Day" event, NTNU, Trondheim, Norway, Dec 2018.

A.2 Recent publications

1-*A.Sinaeepourfard*, J. Krogstie, T. Soltvedt, T. Skuggevik, "*Large-Scale Information and Communications Technology (ICT) Management in Smart Cities based on Edge to Cloud*

Orchestration", IEEE International Conference on Omni-layer Intelligent systems (COINS 2020), Barcelona, Spain, September 2020.

2-*A.Sinaeepourfard*, S.Sengupta, J.Krogstie, R. Ruiz Delgado, "*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 (IINTEC 2019), Hammamet, Tunisia, December 2019.

3-*A.Sinaeepourfard*, J.Krogstie, SA.Petersen, "*A Distributed-to-Centralized Smart Technology Management (D2C-STM) model for Smart Cities: a Use Case in the Zero Emission Neighborhoods*", The Fifth IEEE Annual International Smart Cities Conference (ISC2 2019), Casablanca, Morocco, October 2019.

4-*A.Sinaeepourfard*, SA.Petersen, D.Ahlers, "D2C-SM: Designing a Distributed-to-Centralized Software Management Architecture for Smart Cities", 18th IFIP Conference on e-Business, e-Services and e-Society (I3E 2019), Trondheim, Norway, September 2019.

5-*A.Sinaeepourfard*, J.Krogstie, SA.Petersen, "D2C-DM: Distributed-to-Centralized Data Management for Smart Cities based on two ongoing case studies", Intelligent Systems Conference (IntelliSys) 2019, London, UK, September 2019.

6-*A.Sinaeepourfard*, SA.Petersen, "*Distributed-to-Centralized Data Management through Data LifeCycle Models for Zero Emission Neighborhoods*", TopHPC (High-Performance Computing), Tehran, Iran, April 2019.

7-*A.Sinaeepourfard*, J.Krogstie, SA.Petersen, D. Ahlers, "F2c2C-DM: A Fog-to-cloudlet-to-Cloud Data Management Architecture in Smart City", IEEE 5th World Forum on Internet of Things (IEEE WF-IoT 2019), Limerick, Ireland, April 2019.

8-*A.Sinaeepourfard*, J.Krogstie, SA.Petersen, "A Big Data Management Architecture for Smart Cities based on Fog-to-Cloud Data Management Architecture", NOBIDS 2018, Trondheim, Norway, November 2018.

9-*A.Sinaeepourfard*, J.Krogstie, SA.Petersen, A.Gustavsen, "A Zero Emission Neighborhood Data Management Architecture for Smart City Scenario: Discussions Toward 6Vs Challenge", IEEE ICTC 2018, South of Korea, October 2018.

A.3 Recent activity and dissemination of the results

1-"Workshop Organizer and Idea Creator" of "The First International Workshop on (3SCity-E2C) Building Software Services in Smart City through Edge-to-Cloud orchestration" in conjunction with "The 21the IEEE International Conference on Mobile Data Management (IEEE MDM2020)."

2-IDI, NTNU Grant for organizing the "An ICT in Smart City Day event", Microsoft company and SINTEF are a collaborator in the event, Trondheim, Norway, October 2019.

3-"Session Chair on three different sessions" in IEEE WF-IoT 2019, Ireland, April 2019.

- TP3-1: IoT System Interfaces
- TP1-7: Data Storage and Management for IoT
- TP2-8: Connectivity for IoT

4-DION Grant Winner for organizing the "Smart City Day" event, NTNU, Trondheim, Norway, Dec 2018.

5-"Session Chair on Smart City session" in IEEE ICTC 2018, South of Korea, October 2018.

A.4 Recent Supervision/Co-Supervision Activities

Master Students:

1-Ricardo Daniel Ruiz Delgado, Exchange Student from "Carlos III University of Madrid (UC3M), Spain" ("Title: Anomaly detection in large-scale IoT systems"), 2019-2020.

2-Levi Sorum ("Title: An SDN Architecture Proposal for Smart Cities based on Edge-to-Cloud orchestration"), 2019-2020.

3-Mari Fredriksen ("Title: ICT Architecture in Large Scale Smart Cities"), 2019-2020.

4-Fredrik Strupe ("Title: A Distributed-to-Centralized Architectural Model for Smart Cities using Microservices"), 2019-2020.

5-Petter Rostrup (Title: "A Distributed-to-Centralized Architectural model for Smart City Services through Container Orchestration"), 2018-2019. Was implemented with "Python," "Docker Technologies," and "Swarm Computing."

6-Torbjørn Kirkevik Soltvedt (Title: "A Distributed-to-Centralized Cost Model through Service Selection for Smart Cities"), 2018-2019. Was implemented with "JavaScript," "Cloud Services," and "Multi-Attribute Utility Theory (MAUT) algorithm."

7-Thomas Skuggevik, 2018-2019 (Title: "A Distributed-to-Centralized naming and routing mechanism model in Smart Cities"), 2018-2019. Was implemented with "Python," and "SDN technologies."

Bachelor Students:

1- IT2901 Course ID - Group number 17 (Marius Sundnes, Jonas Bjøralt Giske, Bjørn Magnus Valberg Iversen, Nils Christian Danielsen, Katrine Hveding, and Andreas Bergmo Johnsen)

VISION:

«Sustainable neighbourhoods with zero greenhouse gas emissions»