Vrije Universiteit Amsterdam Universiteit van Amsterdam

Master Thesis

# RAHEC: An Edge Computing Reference Architecture for Healthcare

**Author:** Lucas de Geus (2621225)

| | | |
|---|---|---|
| *1st supervisor:* | Lin Wang | |
| *daily supervisor:* | Ninad Joshi | (Deloitte) |
| *2nd reader:* | Henri Bal | |

*A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science*

August 18, 2022

# Abstract

Edge computing is increasingly being used in healthcare systems to cover some of the shortcomings that have been revealed in traditional cloud computing. Moving computing resources closer to the network edge can improve critical medical systems' latency, privacy, bandwidth, scalability, and energy consumption. While this new computing paradigm brings new capabilities and opportunities to the medical domain, it requires an increasing amount of design decisions to be made during the development of systems. Compared to closely related fields like industrial edge computing, the medical domain has attracted less attention when it comes to standardization. As the medical domain requires different quality-of-service requirements, regulations, and evaluation metrics to be considered, it can not be based on architectures from other domains. To this end, we propose RAHEC, a reference architecture for edge computing focused on the medical domain. It provides a high-level overview of edge computing healthcare systems' recommended products, services, and technologies. More in-depth information originates from our three-dimensional reference model, consisting of six layers, types of edges, and cross-layer concerns. It results in 216 views that can be considered in healthcare edge computing systems. To show the added value and possible shortcomings of our proposed artifacts, we implemented a proof-of-concept. It was shown that RAHEC can aid in making design decisions at each layer of a medical system. However, it showed bias towards cloud vendors' products, lack of cross-layer concern consideration for an entire system, and a generic view that causes excessive information to be provided for simple use-cases.

# Acknowledgements

During the process of conducting this research, I received an incredible amount of support.

First of all, I would like to thank my supervisor from VU Amsterdam, Lin Wang, for hearing my challenges and giving me constructive feedback. Besides the assistance with overcoming technical hurdles, I have also received a lot of help with scientific writing and planning.

I would also like to acknowledge the helpfulness of my colleagues at Deloitte. The access to their network of clients and industry experts was incredibly valuable. I feel privileged that I have been able to carry out this research with the help of both academia and industry.

# Contents

# List of Figures

# List of Tables

# Glossary

| | |
|---|---|
| **AR** | Augmented Reality |
| **AWS** | Amazon Web Services |
| **CDN** | Content Delivery Network |
| **EC** | Edge Computing |
| **ETSI** | European Telecommunications Standards Institute |
| **GCP** | Google Cloud Platform |
| **IAM** | Identity and Access Management |
| **IoMT** | Internet of Medical Things |
| **IoT** | Internet of Things |
| **IP** | Intellectual property |
| **KPI** | Key Performance Indicator |
| **LF-Edge** | Linux Foundation Edge |
| **LMN** | Last mile network |
| **MEC** | Multi-access Edge Computing |
| **NIST** | National Institute of Technology |
| **OS** | Operating System |
| **PLC** | Programmable Logic Controller |
| **PoC** | Proof of Concept |
| **QoS** | Quality of Service |
| **RAHEC** | Reference Architecture Healthcare Edge Computing |
| **RAMEC** | Reference Architecture Model Edge Computing |
| **RAMI4.0** | Reference Architecture Model Industry 4.0 |
| **RAN** | Radio Access Network |
| **RTOS** | Real-time Operating System |
| **SLA** | Service Level Agreement |
| **VPC** | Virtual Private Cloud |
| **VPN** | Virtual Private Network |
| **VR** | Virtual Reality |

# GLOSSARY

# 1

# Introduction

The Internet of Things has successfully been used in healthcare to reduce pressure on doctors and nurses (8). It has been deployed in many environments, including homes, hospitals, and offices, increasing healthcare quality and reducing the cost (9). Commercial wearables and IoT sensors have become popular products to help users gain insight into general metrics about their health, for example, by tracing their daily steps, heart health, sleep quality, calories burnt, and stress levels. Deloitte predicts that 320 million wearable devices will be sold in 2022, increasing to 440 yearly sales in 2024 (10). This commercial market is not the only contributor to the so-called Internet of Medical Things, hospitals and other medical facilities have a significant contribution. For example, in the United States, there is an average of 10-to-15 devices with some form of connectivity per patient bed (11).

Cloud computing is often used in IoT healthcare systems to provide additional storage and processing power (12). However, with the whole domain of IoT being predicted to generate 73.1 Zettabyte of data in 2025 and the existence of new Quality of Service (QoS) requirements, some limitations of traditional cloud computing have been revealed (13). Considering the medical domain, one can imagine critical IoT systems that require QoS requirements such as higher availability, response time, and robustness than cloud vendors can offer (14). In recent years, a new computing paradigm has emerged to cover some of these shortcomings of cloud computing, most often named edge computing (15). The fast-growing field of edge computing has gained a lot of academia and industry attraction. IDC predicts that $74 billion will be spent on edge computing hardware, software, and services across all industries by 2025 (16). It has been shown to improve the latency, privacy, bandwidth issues, scalability, and energy consumption of computing systems, which are crucial for critical medical systems.

# 1. INTRODUCTION

The term edge computing is used since the data processing is no longer primarily performed by centralized nodes but also at the extreme of the network (edge). Data processed closer to the IoT device that produces it requires fewer resources from centralized nodes, thereby reducing network traffic and latency. This distributed approach has shown to enable new types of systems that cloud computing solutions find challenging to realize, such as smart cities (17), autonomous vehicles (18), agricultural land management (19) and virtual/augmented reality (VR/AR) (20).

Due to its wide variety of systems, multiple visions of edge computing exist. The three overarching terms that are most common are *edge computing*, *Fog Computing*, and *Multi-access edge computing (MEC)*. Despite several attempts to establish standards for the definitions and architectures of edge computing, a variety of visions is still found in academia and industry. The European Telecommunications Standards Institute (ETSI) has provided a MEC reference architecture (21) that got adopted by many large companies such as Vodafone, Apple, Samsung, and Intel[1]. However, being limited to MEC, it does not cover all topics that are related to the field of edge computing. A more complete and open-source lexicon of definitions is provided by the Linux Foundation[2].

The variety of systems causes the entire stack (hardware to software) used in edge computing systems to be very heterogeneous. EC systems come in many forms, from a system running solely on an embedded device to a larger system sending data from an embedded device to the cloud via an edge node. This variety and the lack of available reference architectures for use cases causes developers of such systems to make many architectural trade-offs instead of focusing on developing the actual system. Besides the technical challenges a developer faces, the evaluation of EC systems is non-trivial due to the vast amount of different devices involved (22). Developers focused on healthcare EC systems need to deal with additional complications caused by regulation, such as the GDPR[3] for European Union and the HIPPA[4] for USA. To the best of our knowledge, no existing works consider the regulation and required privacy for healthcare systems in their reference architectures.

To this end, this study proposes a reference architecture for developing edge computing systems in the healthcare domain. The reference architecture aims to provide a broader view than technical advice, including guidelines on the evaluation and compliance with the regulation of edge computing healthcare systems. The main research question (RQ) and two sub-questions (SQ) of this study are defined as follows:

[1]https://portal.etsi.org/TB-SiteMap/MEC/List-of-Members
[2]https://github.com/State-of-the-Edge/glossary/blob/master/edge-glossary.md
[3]https://eur-lex.europa.eu/eli/reg/2016/679/oj
[4]https://www.hhs.gov/hipaa/for-professionals/privacy/index.html

- How should we architect an edge computing system for healthcare systems? (RQ)
- How can edge computing systems for healthcare comply with law and regulation? (SQ1)
- How can edge computing systems for healthcare be evaluated objectively? (SQ2)

The remainder of this study is organized in the following manner. Chapter 2 will introduce the concepts of reference architecture, edge computing, and related paradigms. Furthermore, it discusses the edge computing taxonomy adopted in this study. Chapter 3 will cover related work on edge computing and healthcare architectures, discussing their value and how they differ from this study. Next, our proposed reference model and architecture are explained in Chapter 4. To show the practical use of this study, a proof of concept was made using the reference architecture and elaborated on in Chapter 5. Chapter 6 answers the formulated research questions and provides open challenges for research in healthcare edge computing.

# 2

# Background information

In this section, we give an overview of the key concepts to understand the characteristics of the fields of edge computing for healthcare and reference architectures. First, the general concept of a reference architecture is explained and its purpose is motivated. Secondly, a taxonomy by the Linux Foundation Edge (LF-Edge) is described. Afterwards, important terms frequently occurring in related literature are briefly explained and classified within the taxonomy when applicable.

## 2.1 Reference architecture

A *Reference Architecture* is a document or set of documents that provides recommended structures, products and services to form a solution (23). It contains industry best practices, usually to optimise the delivery method for specific technologies. Reference architectures can aid managers, developers, designers and other IT-related professionals in collaborating and communicating about the implementation of a project[1]. The result is that common problems and questions can be solved more rapidly using the proposed solutions. ISO/IEC/IEEE defines a standardised method of creating architectural viewpoints, frameworks and descriptions in the 42010:2011 standard (24).

## 2.2 Edge computing

The origin of edge computing goes back as far as the 1990s when Akamai introduced its content delivery network (CDN) (25). It used nodes that were placed closer to the end-user. In more recent years, the interest in edge computing started growing dramatically

[1]https://www.hpe.com/us/en/what-is/reference-architecture.html

due to the rise of mobile computing and IoT, and many attempts to establish a standard have been made. In 2013, Nokia and IBM introduced RCAS (Radio Applications Cloud Server), which is seen as the first mobile edge computing platform. The year after, ETSI (21) started a standardisation effort for mobile edge computing, followed up a year later by the OpenEdgeComputing (OEC). OEC was a collaboration between Carnegie Mellon University and a handful of important companies (Huawei, Intel, and Vodafone), aiming to shape the global edge computing ecosystem. The initiative still exists and has gained more attraction from important organisations in the industry, such as Amazon and Microsoft[1]. An initiative driven by academia and industry is the IEEE/ACM Symposium on Edge Computing[2], which was first held in 2016. It forms a sound basis for finding state-of-the-art research and applications of edge computing as a proceeding is published annually, showcasing important works in the field.

Despite the efforts for standardisation, there is still no clear definition. Terms such as fog computing, mist computing, cloudlets, and edge computing are used interchangeably in academia and industry. In this study, we have adopted the formal taxonomy provided by the LF-Edge (1), as it covers the whole range of 'edge devices', from centralised data centers to embedded IoT devices. It has recently gotten adoption by large organisations such as ARM, Intel, Samsung, DELL, and VMWare. Figure 2.1 summarizes the taxonomy. Edge computing as a whole is seen as computing capabilities at the logical extreme of a network. The taxonomy divides the *edge* into many subcategories. The first division is made at the last mile network, between the *User Edge* and the *Service Provider Edge*. The last mile network (LMN) connects the service providers, such as telecommunication networks, to the infrastructure of end-users.

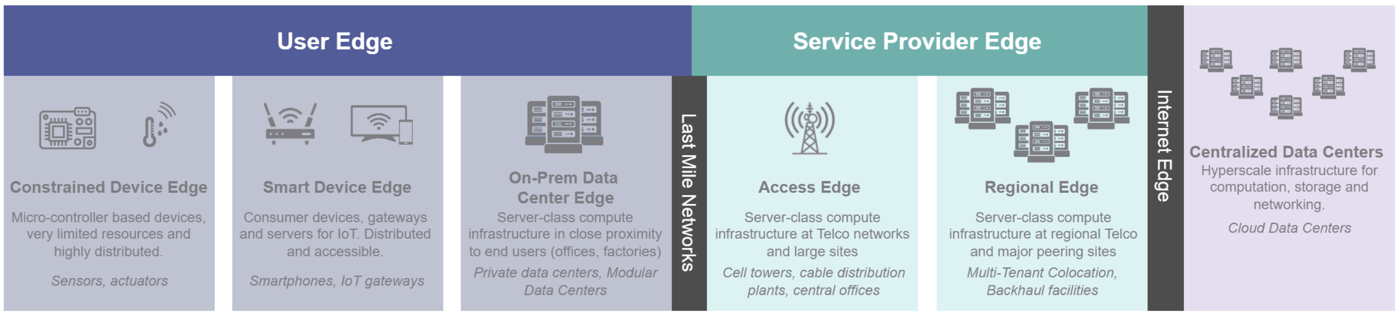

User Edge
Service Provider Edge
Constrained Device Edge
Micro-controller based devices, very limited resources and highly distributed.
Sensors, actuators
Smart Device Edge
Consumer devices, gateways and servers for IoT. Distributed and accessible.
Smartphones, IoT gateways
On-Prem Data Center Edge
Server-class compute infrastructure in close proximity to end users (offices, factories)
Private data centers, Modular Data Centers
Last Mile Networks
Access Edge
Server-class compute infrastructure at Telco networks and large sites
Cell towers, cable distribution plants, central offices
Regional Edge
Server-class compute infrastructure at regional Telco and major peering sites
Multi-Tenant Colocation, Backhaul facilities
Internet Edge
Centralized Data Centers
Hyperscale infrastructure for computation, storage and networking.
Cloud Data Centers

**Figure 2.1:** Taxonomy of Edge Computing by LF-Edge (1)

[1]https://www.openedgecomputing.org/oec-members/
[2]https://acm-ieee-sec.org/

The *User Edge* is deployed on the user side of the LMN. The workloads performed typically aim to reduce latency and bandwidth by processing data near the end-device instead of passing it across the LMN. It contains three subcategories: 1) *Constrained Device Edge* consisting of micro-controller-based devices with very limited resources and can be highly distributed geographically. Devices range from simple sensors and actuators to more powerful devices such as Programmable Logic Controllers (PLCs). 2) *Smart Device Edge* includes mobile devices, personal computers, gateways, and servers that are resistant to challenging environments such as patient homes and outdoor areas. Devices can be stand-alone or embedded and are capable of general-purpose computing but limited in resources such as battery and cost. 3) *On-Prem Data Center Edge* consists of servers located near the end device, such as private data centers in medical facilities.

On the other side of the LMN exists the *Service Provider Edge*, divided into two categories: 1) *Access Edge*, used to connect the *User Edge* with layers in the network that have more compute and storage resources. It is distributed and can be found at common facilities such as cell towers, central offices, radio base stations, and other sites used for physical connections. 2) *Regional Edge* is often located at large peering sites used as multi-tenant co-location facilities and owned by large firms such as IBM, Equinix, and ATT. The sites are typically used by CDNs, cloud providers, and other large-scale web companies to reduce the amount of network hops end users have to take before reaching their data center. It allows large applications to reduce their latency, for example, by caching content, but can not realise the ultra-low-latency required at the *User Edge*, which is sometimes required by critical healthcare applications. Lastly, the *Centralized Data Centers* are the most centralised form of computing within EC systems. The on-demand form of computing and storage is typically used for tasks that allow higher latency and require a significant amount of computing power.

## 2.3 Multi-access Edge Computing (MEC)

Multi-access edge computing, formerly known as Mobile Edge Computing, is a system that provides IT services within the Radio Access Network (RAN) in 4G and 5G, according to the definition of ETSI (21). Although moved closer to the end devices (network edge), MEC services are similar to traditional cloud computing services. ETSI provides an open framework and reference architecture, which allows for easy and efficient integration of applications. MEC is a common term in the field of edge computing, and according to the

ETSI definition and the close relationship to the telecom industry, we classify the access points within the RANs as *Service Provider Edge*.

## 2.4 Internet of (Medical) Things

The Internet of Things consists of physical objects ("things") in the real world that are embedded with sensors and actuators to connect and exchange data with other devices and systems over the internet. Typical characteristics include heterogeneity of hardware and networks, vast amounts of devices and events, and data generated based on "things" in the real-world (23). The Internet of Medical Things (IoMT) is a sub-field of IoT focused only on the medical field. It refers to the medical things that, in some form, acquire biomedical signals and transfer it as data over a network without needing human-to-human interaction. In related literature, Internet of (Medical) Things devices can be mapped to the *User Edge*.

With the heterogeneity of devices in the IoMT, related literature typically makes a distinction into multiple categories similar to the one made by Nanaykkara et al. (26). For clarity, we mapped their categories to the LF-Edge taxonomy. The mappings are shown in table 2.1.

| **IoMT Category** | **Corresponding edge type in taxonomy** | **Example** |
|---|---|---|
| Wearable Sensors/Devices | Constrained Device Edge | Heart rate monitor (HRM) |
| Implantable Sensors/Devices | Constrained Device Edge | Glucose sensor |
| Ambient Sensors/Devices | Constrained Device Edge | Temperature sensor |
| Stationary Sensors/Devices | Smart Device Edge | Magnetic Resonance Imaging (MRI) |

**Table 2.1:** IoMT categories mapped to LF-Edge taxonomy

## 2.5 Terminology

This study aimed to use the terminology proposed in the LF-Edge taxonomy (fig. 2.1). Related literature discussed in this study often does not adhere to this terminology. For that purpose, their terminology is mapped to the LF-Edge taxonomy when it is reasonable. In case related literature uses general terms, such as *Edge Computing* or *Fog Computing*, or the terms can not be reasonably mapped to the LF-Edge taxonomy, we use the original terminology.

When referred to *large cloud vendors*, in this study, we consider Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure. This decision is justified by the fact that the three vendors have a combined market share of >50% (27).

# 3

# Related work

In this section, we analysed related work on the architecture of edge computing systems and related paradigms. A wide variety of reference architectures have been published for specific industries as well as more generic architectures. First, industry initiatives are discussed, followed by related academic work.

ETSI continuously provides and updates a framework and reference architecture for multi-access edge computing (21). It contains detailed guidelines for developing a MEC system that allows MEC applications to run efficiently and seamlessly in a multi-access network. The generic reference architecture shows the functional elements required to create a MEC system and how they are connected. Detailed instructions on implementing virtualisation concepts (Network Functions Virtualisation) in the system were also introduced.

The Edge Computing Consortium (ECC) and Alliance of Industrial Internet (AII) published Edge Computing Reference Architecture 2.0 (28), a model-based architecture with open interfaces at each horizontal layer. It was developed using the ISO/IEC/IEEE 42010:2011 standard, which is broadly used. The layers are *Smart Services*, *Service Fabric*, *Connectivity and Computing Fabric*, and *Edge Computing Node*. However, vertically it considers *management*, *data life-cycle*, and *security*, for the purpose of ensuring smart services throughout the entire system. Several large organisations, including Huawei, Intel, and ARM, jointly created the ECC.

The German Electrical and Electronic Manufacturers' Association developed a more domain-specific architecture. The Reference Architecture Model Industry (RAMI 4.0) (2) provides a framework for developing products and business models within Industry 4.0. It consists of a three-dimensional map that represents the most important aspects of Industry 4.0, shown in fig. 3.1. The three-dimensional approach to identifying important considerations in a specific industry is adopted in this study and other academic works.

Several other architectures have been published by industry-backed organizations, including the FAR-Edge RA (29), INTEL-SAP RA (30), Industrial Internet Consortium RA (31), and IBM RA (32).

Academia has focused on industry-specific frameworks and architectures. Willner and Gotham (33) proposed the Reference Architecture Model Edge Computing, taking a similar approach as the developers of RAMI 4.0 by creating a three-dimensional matrix with five concerns, six layers, and seven levels. It is focused on industrial edge computing, and the result provides 210 views on the edge computing paradigm in the manufacturing domain. The concerns and layers introduced by Willner and Gotham inspired this work. However, the levels (types of edges) differ since we adopted the LF-Edge taxonomy. Another difference is the focus of the field, which was manufacturing for Willner and Gotham, and healthcare in our study.

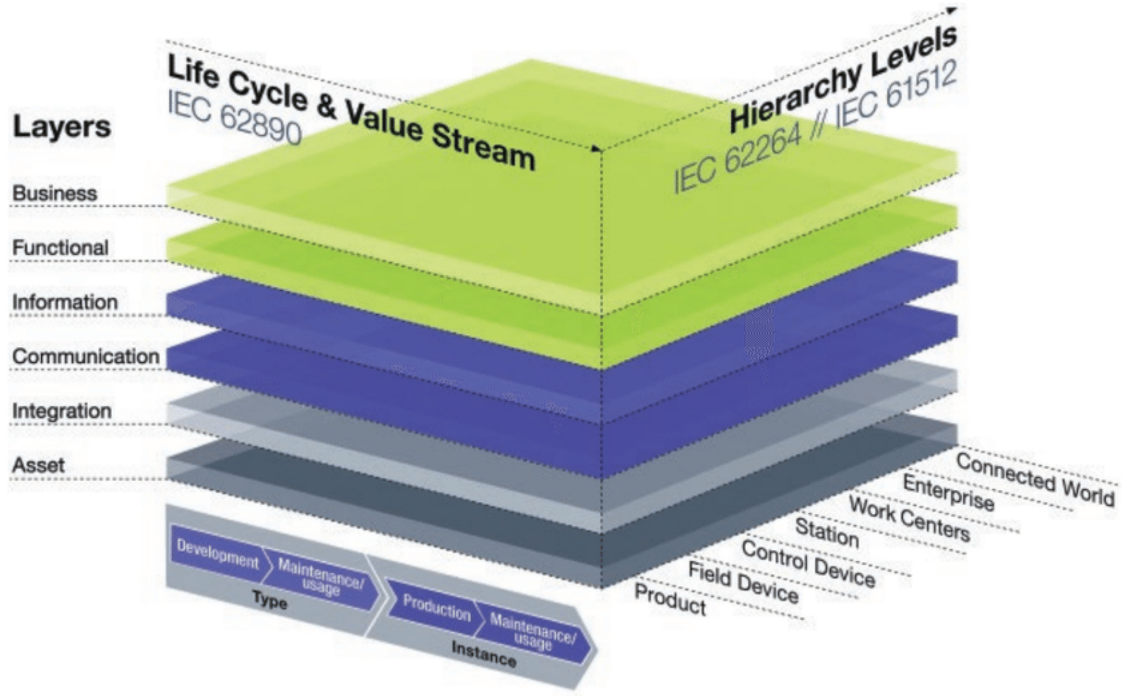


**Figure 3.1:** Reference Architecture Model Industrie (RAMI 4.0) (2)

BodyEdge, an architecture for human-centric applications in the healthcare domain, was proposed by Pace et al. (34). It uses a three-tier architecture (IoMT device - edge - cloud) and has been shown to reduce transmitted data and processing time of healthcare applications. However, since its main focus is on reducing the data traffic towards the internet, it does not provide developers with all information needed for developing critical healthcare applications (e.g. on security and regulation).

Al-khafajiy et al. (35) similarly used a three-tier architecture focused on healthcare. It is focused on optimizing the management of resources and job allocations to achieve a high QoS with regard to latency. The proposed prototype consists of a patient monitoring system that runs optimally when the IoMT device, edge node, and cloud are connected but can also run independently at the edge in case of network failure.

# 4

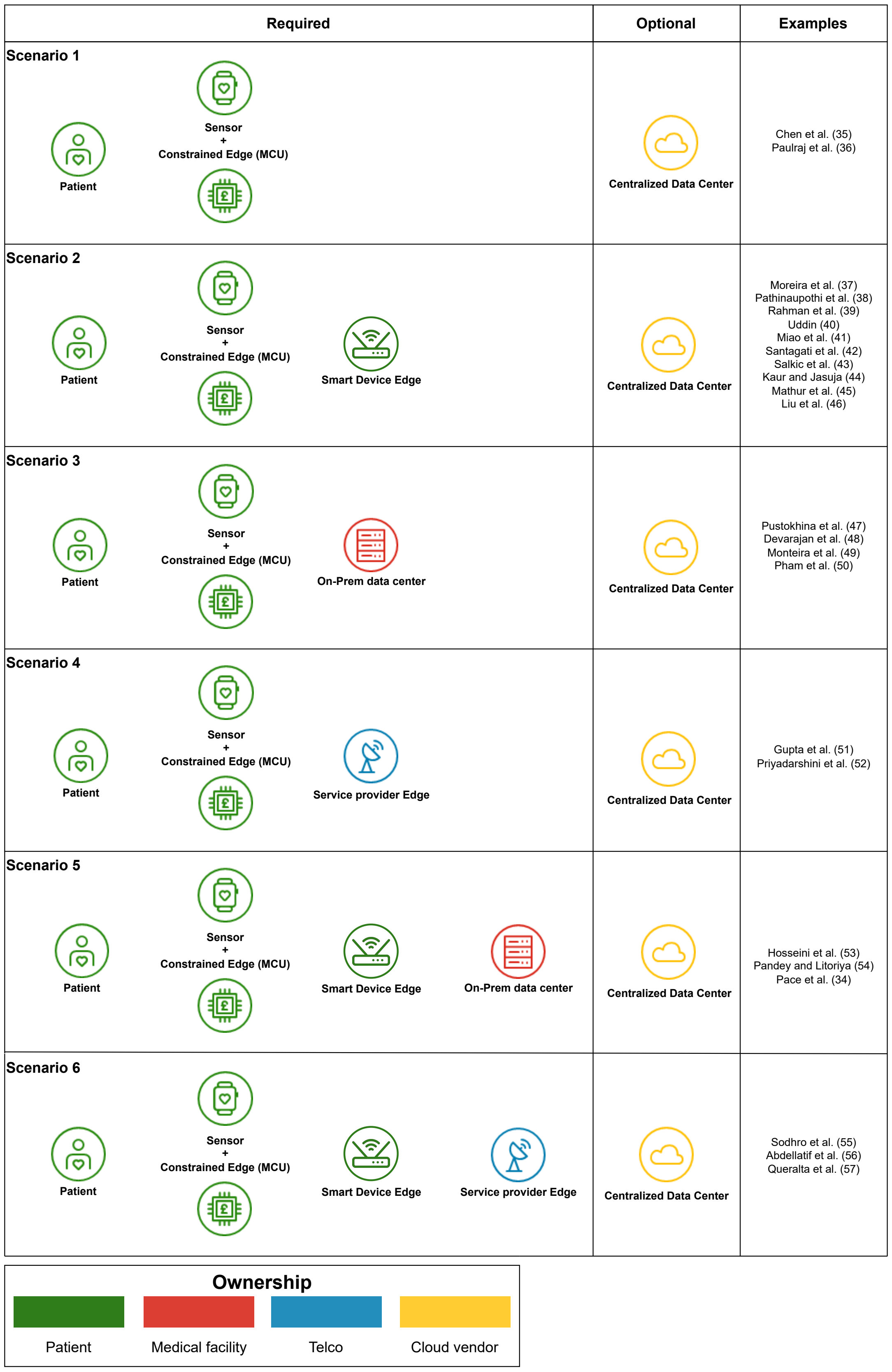
# Reference Model and Architecture

This section discusses the different edge computing healthcare scenarios, proposed reference model, and proposed Reference Architecture Healthcare Edge Computing (RAHEC). To create a reference architecture, we first identify different scenarios from research and industry in which a combination of edges (as defined in fig. 2.1) can be used in healthcare systems. Next, we create a three-dimensional matrix with concerns, layers, and levels which form a reference model. Detailed information and recommendations are given for each layer. The combination of the views that result from the reference model and the possible scenarios form the basis for a generic edge computing healthcare architecture.

## 4.1 Scenarios

Although we have defined six types of edges, typical EC healthcare systems do not use all edges. Figure 4.1 shows the identified scenarios. This study only considers scenarios that use some medical constrained edge device. The ownership shown in the figure is based on the most common scenario. In specific systems, this might deviate, for example, the *On-Prem Data Center* might be owned by the patient in case it is a smart-home server. The examples given in the table are research papers discussing a medical application using edge computing.

Table references: (36) (37) (38) (39) (40) (41) (42) (43) (44) (45) (46) (47) (48) (49) (50) (51) (52) (53) (54) (55) (34) (56) (57) (58)



| Required | Optional | Examples |
|---|---|---|
| **Scenario 1** Patient; Sensor + Constrained Edge (MCU) | Centralized Data Center | Chen et al. (35)<br>Paulraj et al. (36) |
| **Scenario 2** Patient; Sensor + Constrained Edge (MCU); Smart Device Edge | Centralized Data Center | Moreira et al. (37)<br>Pathinaupothi et al. (38)<br>Rahman et al. (39)<br>Uddin (40)<br>Miao et al. (41)<br>Santagati et al. (42)<br>Salkic et al. (43)<br>Kaur and Jasuja (44)<br>Mathur et al. (45)<br>Liu et al. (46) |
| **Scenario 3** Patient; Sensor + Constrained Edge (MCU); On-Prem data center | Centralized Data Center | Pustokhina et al. (47)<br>Devarajan et al. (48)<br>Monteira et al. (49)<br>Pham et al. (50) |
| **Scenario 4** Patient; Sensor + Constrained Edge (MCU); Service provider Edge | Centralized Data Center | Gupta et al. (51)<br>Priyadarshini et al. (52) |
| **Scenario 5** Patient; Sensor + Constrained Edge (MCU); Smart Device Edge; On-Prem data center | Centralized Data Center | Hosseini et al. (53)<br>Pandey and Litoriya (54)<br>Pace et al. (34) |
| **Scenario 6** Patient; Sensor + Constrained Edge (MCU); Smart Device Edge; Service provider Edge | Centralized Data Center | Sodhro et al. (55)<br>Abdellatif et al. (56)<br>Queralta et al. (57) |

**Ownership**

| 🟩 | 🟥 | 🟦 | 🟨 |
|---|---|---|---|
| Patient | Medical facility | Telco | Cloud vendor |

**Figure 4.1:** Edge Computing Healthcare scenarios

## 4.2 Reference model

The edge computing paradigm covers a wide range of topics and technologies. To create a high-level model of what could be considered during the development of a healthcare EC system, we take a similar approach as Willner and Gotham (33). Their model was based on RAMI 4.0, consisting of a three-dimensional map representing the most important aspects of Industry 4.0. Willner and Gowtham adjusted the 3D map to represent the Reference Architecture Model Edge Computing (RAMEC).

Figure 4.2 shows the reference model we propose for the healthcare domain. The *layers* are based on the RAMEC model, as these do not differ between the fields of Industry 4.0 and Healthcare. The *edge types* are based on the LF-Edge taxonomy, covering all possible types of edges in an EC healthcare system. The *cross-layer concerns* are partly based on the RAMEC model. However, additional *concerns* were required for the healthcare domain. Due to the large number of connected devices, *scalability* was important to consider. *Resilience* was added because the systems need to be able to recover or adapt to events that cause the loss of mission-related functions. This is confirmed by research from IDC (59), showing the importance of resilience in healthcare systems and how edge computing can aid with this goal. Additionally, *scalability* and *resilience* are present in the system characteristics of the Industrial Internet of Things Reference Architecture (60). As mentioned before, *evaluation* and *regulation* are important aspects of the medical domain. Therefore, both were considered in this study and added as cross-layer concerns. The result of this model consists of 216 (6x6x6) views that could be considered for healthcare EC systems.

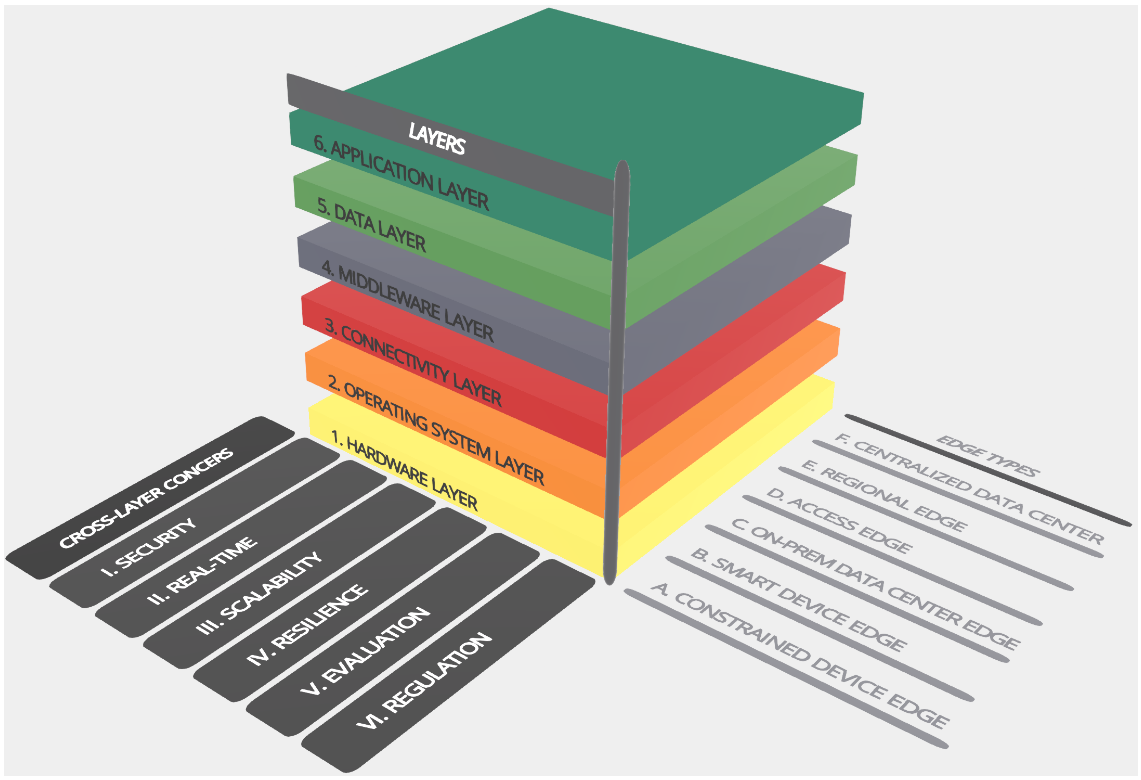


**Figure 4.2:** Reference model: Edge Computing Healthcare

Important to mention is that the views are not always relevant to a developer of healthcare systems. For example, a project team implementing an EC system using a cloud provider is unlikely to consider the view Security-Hardware-CloudCenter, as they do not have any authority over the specific hardware used in traditional cloud centers. Likewise, systems tend to use only a subset of the types of edges. If a system, for example, does not use an On-Prem Data Center Edge, there is no point in considering the views that contain this type of edge.

While layer-specific recommendations are given, layers can influence each other. For example, the required network stack in the connectivity layer influence what hardware and operating systems are required in the system, as they might need to support specific technologies and protocols. Or vice versa, if the budget dictates that the embedded devices at the edge can not contain a Bluetooth module, design decisions in the connectivity layer must not contradict this.

### 4.2.1 Hardware Layer

The hardware used in EC Healthcare systems can be split into two categories. We consider the *User Edge* where the developer has a significant amount of influence on the design and can customize everything, and the *Service Provider Edge and Data Center* which is often provisioned on demand and consists of a broad (but limited) range of hardware.

#### 4.2.1.1 Service Provider Edge and Data Center

When using traditional *Centralized Data Centers*, developers must consider different types of available hardware. The main choices that must be considered are instance types and storage options. Large cloud vendors typically offer different hardware for specific problems, for example, CPU-focused machines for general purpose tasks, GPUs for heavy graphical tasks, and SSDs for heavy I/Os. Clear documentation with explanations and decisions trees on which instance type is most suitable are given based on different use cases[1] [2] [3].

In recent years, cloud vendors have picked up on the trend of edge computing and started providing Infrastructure as a Service closer to the edge of the network. These so-called "edge servers" typically contain a subset of the hardware available in the traditional *Centralized Data Centers*. Well-known examples include AWS Local Zones, AWS Wavelength, Azure Public Multi-Access Edge Compute, and Google Distributed Cloud Edge. To bring their resources closer to the edge, cloud vendors typically partner with communication service providers (e.g. Vodafone and Verizon for AWS Wavelength) by using their 5G networks that provide low latency and high bandwidth, resulting in the *Service Provider Edge*.

Most *cross-layer concerns* (*security, real-time, scalability, and resilience*) are covered in the Service Level Agreement (SLA) and policies provided by cloud vendors. The most common measurement is the up-time of different instance types. The *evaluation* of the hardware is use-case dependent, but typically monitoring solutions that allow developers to check KPIs related to the hardware are provided. Such metrics may include CPU and memory utilization and disk traffic metrics. *Regulation* on service providers' hardware is less relevant to developers as they have little influence. However, it is important to consider with which regulations cloud vendors comply. The information on compliance is often provided by the vendor per service.

[1]https://aws.amazon.com/ec2/instance-types/
[2]https://azure.microsoft.com/en-us/pricing/details/virtual-machines/series/
[3]https://cloud.google.com/compute/docs/machine-types

#### 4.2.1.2 User Edge

*User Edge* hardware is important to consider for developers as it defines the technical capabilities the system has in the last mile network. We consider the three types of edges separately.

**On-Prem Data Center Edge** include micro-data centers, for example, located in hospitals, typically consisting of general purpose server hardware complemented by accelerators. The cross-layer concerns are similar to those of the Service Provider Edge, except that the micro-data center tends to be owned by the hospital (or other medical facilities).

**Smart Device Edge** has limited capabilities for various reasons, such as cost, battery life, and form factor. According to the LF-Edge, these devices should maintain a small memory footprint while still having enough resources to perform simple computing tasks at the edge. The minimum memory they tend to have is 256MB. The hardware requirements span from simple mobile devices to IoT gateway servers. Devices may contain accelerators such as GPUs, FPGAs and TPUs for specific tasks such as inferencing at the edge.

Cross-layer concerns are highly relevant for developers, as the flexibility of the hardware in the Smart Device Edge leads to many design decisions that can influence *Security*, *Real-time*, *Scalability*, and *Resilience*. The *evaluation* of the Smart Device Edge is complicated as monitoring the hardware in already resource-limited devices can lead to overhead. With the diversity in available off-the-shelf devices, *regulation* is important to consider. Even though regulation mostly applies to the *Application* and *Data* layers, hardware can be forbidden in regions. A well-known example is the order of the former US president Donald Trump that forbids companies based in the US to collaborate or buy equipment from manufacturer Huawei[1]. Among other things, Huawei produces smartphones that would fit well within the characteristics of a Smart Device Edge. Developers should, therefore, consider not only the technical requirements of their devices but also the possible legal restrictions.

**Constrained Device Edge** forms the basis of the edge computing systems as the devices fetch real-world data using sensors and perform real-world tasks using actuators. Microcontrollers, which can be seen as a computer system-on-a-chip, are typically the heart of these devices. The memory available typically ranges from KBs to a few MBs. According to NXP, a semiconductor designer and manufacturer, the devices must balance between

[1]https://www.whitehouse.gov/presidential-actions/executive-order-securing-information-communications-technology-services-supply-chain/

five different design attributes, shown in fig. 4.3. Improving one attribute leads to the decrease of another. Defining the system requirements beforehand is important to prevent a mismatch between hardware and system capabilities. A recent study by Ali et al. (61) compared different IoT hardware platforms, which are suitable for edge computing, along with detailed information on their processor, GPU, clock speed, memory, communication, IDE, I/O, programming language, and cost.

Most *security* implementations will take place in other layers, but as the *Constrained Device Edge* is typically distributed in the real world and might be exposed to unwanted parties, hardware security is important. External interfaces and ports should be blocked or restricted to prevent malicious usage of the devices. The hardware should also support the partitioning of trusted and untrusted software, which most hardware vendors producing microcontrollers support. Examples of such extensions that support these so-called enclaves are Intel SGX[1], RISC-v PMP[2], AMD SEV[3], and Arm Trustzone[4].

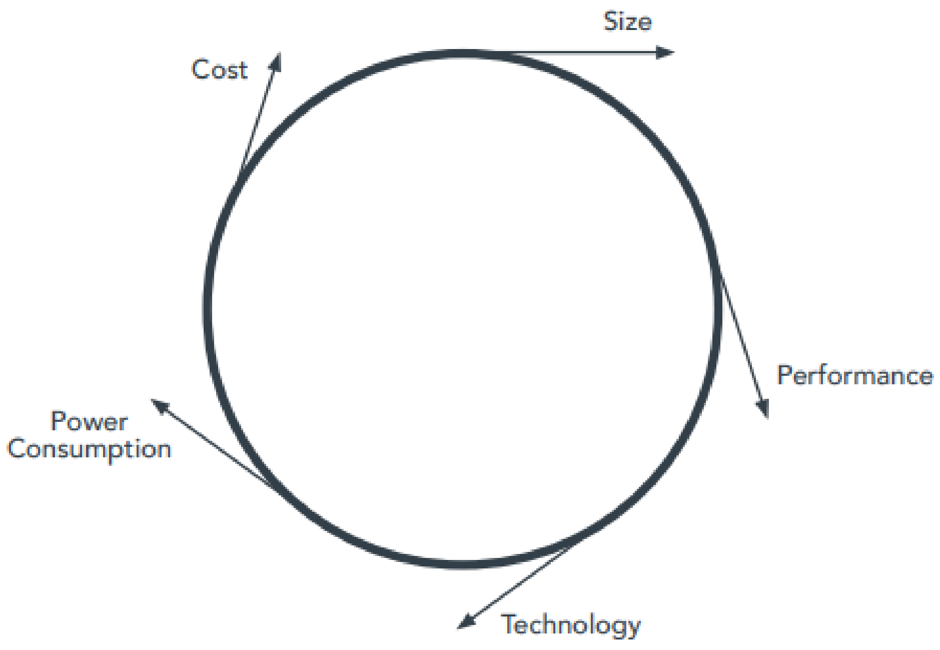


**Figure 4.3:** Parameters that control embedded system success according to NXP (3)

[1]https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html

[2]https://sifive.github.io/freedom-metal-docs/devguide/pmps.html

[3]https://developer.amd.com/sev/

[4]https://www.arm.com/technologies/trustzone-for-cortex-a

### 4.2.2 Operating System Layer

The operating system (OS) that is used throughout different types of edges plays an important role in edge computing systems as it has a large impact on many of the cross-layer concerns such as *Security*, *Real-time*, *Scalability*, and *Resilience*. Different types of edges require different operating systems, which are discussed separately.

#### 4.2.2.1 Service Provider Edge and Data Center

Operating systems are limited by what the cloud vendors offer. The large vendors typically support a wide range of Linux distributions (e.g. CentOS, Debian, Fedora, RedHat Enterprise, and Ubuntu Server) and different Windows Servers. In general, determining the best OS is use case dependent, but Linux distributions are most common. Overviews of the characteristics of supported operating systems are provided by cloud vendors, which can guide developers. Google Cloud Platform, for example, provides information on the cost, security, user space, networking, and support for different operating systems[1].

#### 4.2.2.2 On-Prem Data Center Edge

*On-Prem data centers* can run most OS. Like the Service Provider Edge, the choice is usually a Linux distribution or Windows Server. Typically the OS is chosen based on the applications that will run on it. A management platform is usually deployed on-premise to create a hybrid cloud environment that allows integration with the cloud. These management platforms require specific operating systems. For example, to run Google Anthos clusters on-premise, only CentOS, RedHat Enterprise Linux, and Ubuntu are allowed.

#### 4.2.2.3 Smart Device Edges

*Smart Devices* can contain a wide variety of hardware and therefore operating systems. The OS depends on the device's capability and the use case. The most common are Windows, Linux, and mobile OS. The connectivity should be considered when choosing an OS, as the supported networking can be influenced.

[1]https://cloud.google.com/compute/docs/images/os-details#vm-manager

#### 4.2.2.4 Constrained Device Edge

*Constrained Devices* are the most heterogeneous regarding operating systems. With minimal capacity available, the devices need to ensure that the cross-layer concerns are covered. To realise real-time requirements that come with building critical EC healthcare systems, the devices running on small microcontrollers use Real-Time Operating Systems (RTOS). RTOSs can be divided into two categories, hard and soft real-time (62). Hard real-time handles deadlines strictly, meaning that tasks must start and end within a specific time-frame, while soft real-time allows a small delay for deadlines. Healthcare systems that are medical critical often require hard real-time operating systems.

There is a wide variety of open-source and commercial RTOSs available. Table 4.1 provides an overview of some of the most popular OSs for resource-limited devices and some basic characteristics. It is important to note that the given supported hardware vendors numbers refer to the number of vendors of which at least one processor family is supported, and only the vendors mentioned in the official documentation are counted, meaning that more hardware might be supported. Similarly, the given layer 5,6,7 protocols refer to the protocols mentioned and supported by the documentation of the RTOS, meaning that implementing other protocols may be possible. When choosing an RTOS it is important to consider the other layers from the reference model (fig. 4.2). Due to memory availability, processing power and vendor support, the hardware dictates what RTOS can run on the embedded devices. The connectivity layer might require specific technologies and protocols such as UWB, NFC, and Bluetooth, which the RTOS should properly support. Typically an RTOS should have all middleware and drivers required for an embedded system since adding additional middleware can lead to overhead.

| **Name** | **Paid** | **Publisher** | **Supported hardware vendors** | **Layer 5,6,7 protocols** |
|---|---|---|---|---|
| FreeRTOS | No | Amazon | >20 Link | **HTTP, DNS, FTP, TELNET, MQTT** |
| AzureRTOS | No | Microsoft | >20 Link | **HTTP, DNS, FTP, TELNET, SMTP, POP3, MQTT** |
| Zephyr | No | Linux Foundation | >20 Link | **HTTP, DNS, TELNET, MQTT** |
| RIOT-OS | No | RIOT-OS | >20 Link | **HTTP, DNS, TELNET, MQTT** |
| TI-RTOS | No | Texas Instruments | 3 Link | **HTTP, FTP, MQTT** |
| Keil RTX | No | Arm/SoftBank | >20 Link | **N/A** |
| MQX | No | NXP | 1 Link | **HTTP, DNS, FTP, SMTP, MQTT** |
| SafeRTOS | No | Wittenstein HIS | 13 Link | **HTTP, DNS, MQTT** |
| Integrity | Yes | Green Hills Software | 17 Link | **HTTP, DNS,** |
| LynxOS | Yes | Lynx Software Technologies | N/A Link | **HTTP, DNS, FTP** |
| PikeOS | Yes | Sysgo | >20 Link | **MQTT, N/A** |
| VxWorks | Yes | WindRiver | >20 Link | **HTTP, DNS, FTP, TELNET, SMTP, POP3, MQTT** |

**Table 4.1:** Popular Real-time Operating Systems and their characteristics

It should be considered that large cloud vendors often have their own RTOS, which integrates smoothly with their other services. Amazon Web Services provides FreeRTOS, which is connected to AWS IoT Core and AWS IoT Greengrass. Azure RTOS is Microsoft's solution, connecting seamlessly with Azure IoT. Other cloud vendors that have not created their RTOS provide different solutions, such as Google's software development kit for IoT devices, which can be ported to several RTOSs (Zephyr, Arm MBed OS, FreeRTOS).

While requiring considerably more memory and being a soft real-time solution, embedded Linux is noteworthy as it is used for many constrained devices (63). Open-source (Yocto, Buildroot, OpenWrt, Linaro ARM) and commercial (Wind River, FlexOS, OmniOS, Ubuntu Core) embedded Linux distributions are available.

### 4.2.3 Connectivity Layer

Figure 4.4 shows an overview of different connectivity technologies used throughout the edge computing ecosystem. It is based on the *Wireless Connectivity Overview* of NXP (3), with changes made to fit the LF-Edge taxonomy. Besides, several technologies and protocols were added based on other related works. The connectivity technologies for the most common communication scenarios based on fig. 4.1 are considered in this section.

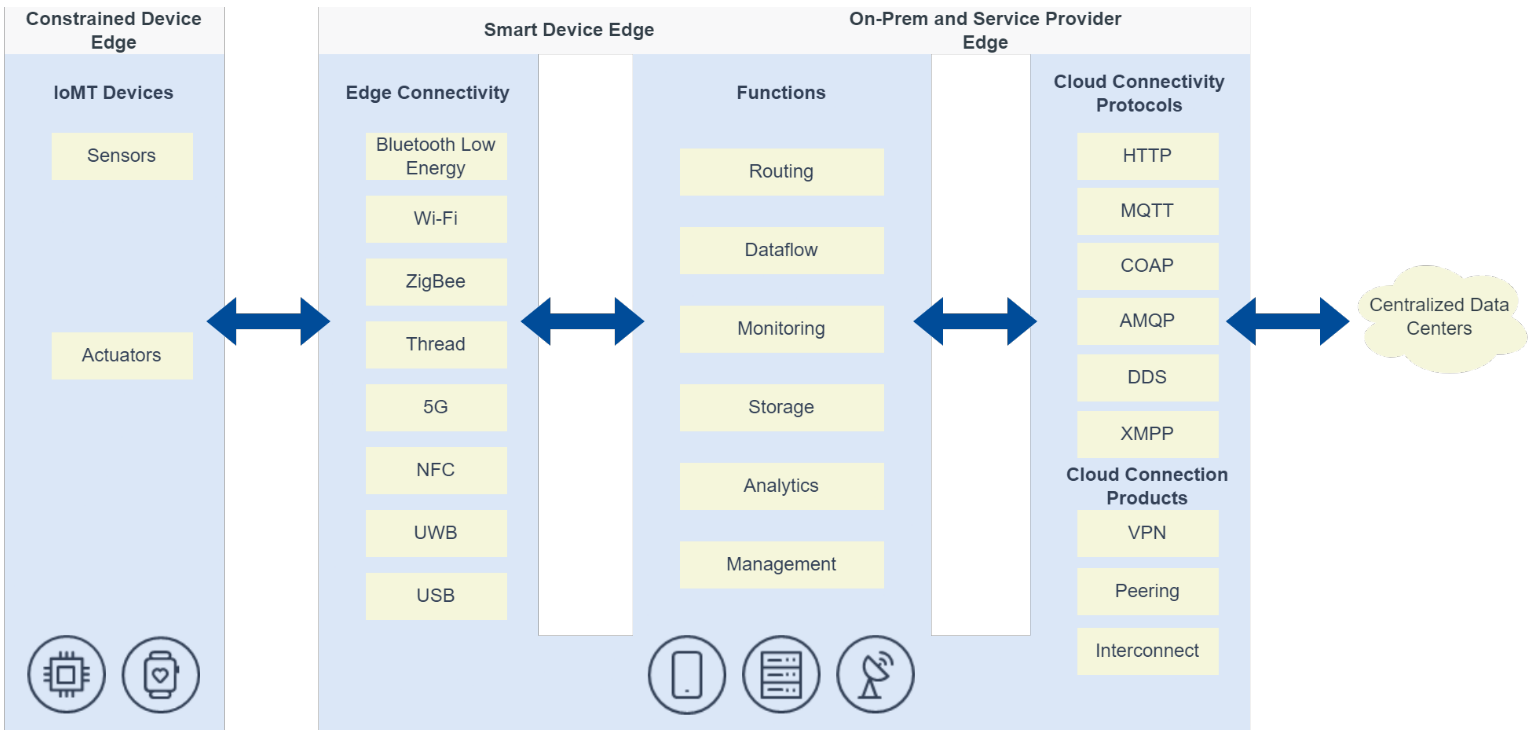


**Figure 4.4:** Connectivity ecosystem in edge computing

#### 4.2.3.1 Service Provider Edge $\Longleftrightarrow$ Centralized Data Center

The communication between the *Service Provider Edge* and *Centralized Data centers* is often taken care of by the cloud vendors themselves, assuming that both are deployed at the same vendor. By including the *SPE* in a Virtual Private Cloud (VPC), the resources can be used as if they are part of the traditional cloud. Resources within the VPC typically communicate using internal IP addresses, making use of TCP. Cloud vendors allow deployments from other vendors to be included in the VPC using a site-to-site VPN.

#### 4.2.3.2 On-Prem Data Center $\Longleftrightarrow$ Centralized Data Center / Service Provider Edge

Different options exist to connect *On-Prem Data Centers* to cloud and telco infrastructure. Developers can use regular protocols that are suitable for cloud connectivity, such as HTTP, MQTT, COAP, AMQP, and DDS. Even though these can be used over the public internet, cloud vendors provide other solutions to ensure more secure and faster connections. Table 4.2 shows the products that AWS, Azure, and GCP provide to connect on-premise infrastructure with their infrastructure securely. The products from GCP have been used as an example in the table, and the other two cloud vendors' products have been mapped accordingly.

| **Type** | **Summary** | **Azure/Aws** | **Use case** |
|---|---|---|---|
| VPN | Virtual Private Network, makes a secure connection between networks over public internet by encrypting all traffic (IPsec protocol). | AWS Client VPN<br>Azure VPN | If using the public internet is acceptable. |
| Direct Peering | Placing a router in the same public data center as a cloud vendor to realise a secure and high bandwidth connection. | AWS Private Peering<br>Microsoft Peering | If using public internet is unacceptable or bandwidth is not reliable enough. |
| Carrier Peering | Placing a router in the same data center as a partner of the cloud vendor to realise a secure and high bandwidth connection. | AWS Public Peering<br>Microsoft Peering | Same as direct peering, but in case the cloud has no point of presence near the on-premise location. |
| Dedicated Interconnect | Extending on-premise network to VPC networks from the cloud provider via a dedicated connection. | AWS Direct Connect<br>Azure ExpressRoute Direct | If high bandwidth and flexible scaling are required, and the public internet is not safe enough. The main difference with Peering is that it requires the use of Google Cloud and has maintenance costs. |
| Partner Interconnect | Extending on-premise network to VPC networks from the cloud provider via a service provider connection. | AWS Direct Connect Delivery Partner<br>Azure ExpressRoute Partner | Same as a dedicated interconnect, but in case the cloud vendor has no point of presence nearby the on-premise location. |

**Table 4.2:** On-premise to cloud infrastructure connectivity products.

#### 4.2.3.3 Smart/Constrained Device Edge $\Longleftrightarrow$ Service Provider Edge / Centralized Data Center

For device-to-cloud communication, we built upon the work of Dizdarević et al. (4) who studied the communication protocols for IoT to fog and cloud computing integration. Two popular communication models are considered: the request-reply and the publish-subscribe. In a system with many devices in the *User Edge* and only a single (or few) endpoints in the *Service Provider Edge* or *Cloud* the publish-subscribe model brings benefits over the request-reply model. Dizdarević et al. identified three benefits: 1) the existence of publishers and subscribers is irrelevant to the involved parties, 2) subscribers can receive information from many publishers, and a publisher can send information to many different subscribers (many-to-many communication), 3) publishers and subscribers do not need a constant connection because the broker can store messages until a client is reconnected.

Table 4.3 shows the different application layer protocols and their compatibility with different clouds. The table shows a sign of adoption towards HTTP and MQTT, this is backed up by a survey (63) conducted by the Eclipse Foundation, showing that HTTP(S) and MQTT are the preferred communication protocols for IoT applications. In IoT systems, MQTT has shown to have lower latency (64) (65), bandwidth (66) and energy consumption compared (67) to HTTP. The security of both protocols relies on TLS/SSL, requiring a handshaking process between the client/server before any further communication can occur. Therefore, in general, MQTT is the advisable protocol, and HTTP should only be used in specific use cases, such as lack of support by the edge device for MQTT. While TLS/SSL is considered safe, many security threats and challenges exist for connectivity. The threats, challenges, and possible mitigation techniques are further discussed in section 4.2.6.

| **Protocol** | **Req-Rep** | **Pub-Sub** | **Standard** | **Transport** | **Security** | **GCP** | **Azure** | **AWS** |
|---|---|---|---|---|---|---|---|---|
| REST HTTP (/2.0) | ✓ | ✓ | IETF | TCP | TLS/SSL | ✓ | ✓ | ✓ |
| MQTT | X | ✓ | OASIS | TCP | TLS/SSL | ✓ | ✓ | ✓ |
| CoAP | ✓ | ✓ | IETF | UDP | DTLS | X | X | X |
| AMQP | ✓ | ✓ | OASIS | TCP | TLS/SSL | X | ✓ | X |
| DDS | X | ✓ | OMG | TCP/UDP | TLS/DTLS/DDS | X | X | X |
| XMPP | ✓ | ✓ | IETF | TCP | TLS/SSL | X | X | X |

**Table 4.3:** Application Layer Protocols features (based on Dizdarević et al. (4)) and cloud support

The *Constrained Edge Devices* are not always capable of directly communicating with the cloud using MQTT/HTTP protocols as they typically lack modules for 3/4/5-G and Wi-Fi, and often rely on a *Smart Device Edge* to function as a gateway. The constrained devices can subscribe and publish to MQTT topics through a gateway device, for example, using a Bluetooth low-energy proxy device (e.g. smartphone).

#### 4.2.3.4 Constrained Device Edge $\iff$ Smart Device Edge

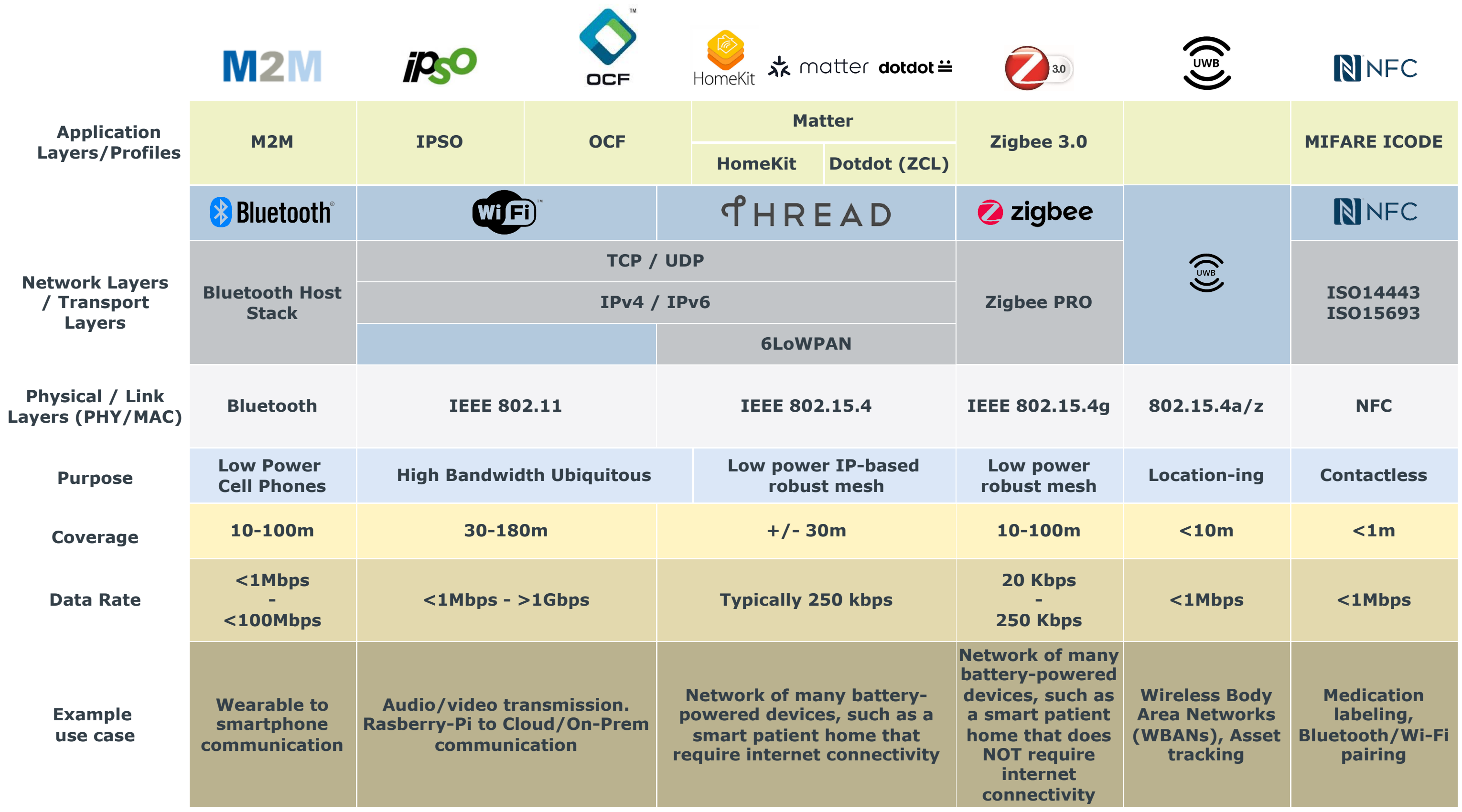
The communication between *embedded devices* and *smart devices* highly depends on the use case. Typically, multiple sensors collect data and send it to one or more *smart devices* that function as a gateway, the gateway is responsible for aggregating the data and, if necessary communicating with the cloud to perform more computing or storage intensive tasks. The design decision for a specific communication technology is often based on the physical distance between the devices, the acceptable power consumption, and the required throughput. Figure 4.5 shows the connectivity software stack mapped approximately as OSI layers according to NXP (3). Additionally, the figure shows the data rates and coverage of the different technologies.

For resource-constrained devices, we focus on *Bluetooth low-energy* (BLE), a highly adopted technology for sending and receiving data in short-range networks. The low power consumption is possible because of the low latency, duty cycle and data-packet size compared to the other Bluetooth types. It has several security features, such as countering eavesdropping, device tracking, and man-in-the-middle attacks. It is typically used for communication between the *Constrained Device Edge* and a *Smart Device Edge* that is used as a gateway to the cloud.

*Wifi-6* can deliver low latency with a long-range and support power-saving techniques for embedded devices. Several protocols such as WPA3, WPA3-Personal, and WPA3-Enterprise allow for a saver network at the edge. It is the preferred technology to connect the *User Edge* to the cloud (3). It is unsuitable for battery-powered devices since the energy consumption is higher than other technologies.

*Thread* is an open-source networking protocol specifically built for IoT[1]. It allows low-power and low-latency device-to-device, device-to-mobile, and device-to-cloud communication. Within a thread network, there are typically many devices present, and due to the network topology, it can be designed to have no single point of failure. Due to its low energy consumption, it is the preferred technology for networks that include many

[1]https://www.threadgroup.org/What-is-Thread/Thread-Benefits



| | M2M | IPSO | OCF | HomeKit | matter dotdot | Zigbee 3.0 | UWB | NFC |
|---|---|---|---|---|---|---|---|---|
| Application Layers/Profiles | M2M | IPSO | OCF | Matter | Matter | Zigbee 3.0 | | MIFARE ICODE |
| | | | | HomeKit | Dotdot (ZCL) | | | |

| | Bluetooth | Wi Fi | THREAD | zigbee | UWB | NFC |
|---|---|---|---|---|---|---|
| Network Layers / Transport Layers | Bluetooth Host Stack | TCP / UDP | TCP / UDP | Zigbee PRO | UWB | ISO14443 ISO15693 |
| | | IPv4 / IPv6 | IPv4 / IPv6 | | | |
| | | | 6LoWPAN | | | |
| Physical / Link Layers (PHY/MAC) | Bluetooth | IEEE 802.11 | IEEE 802.15.4 | IEEE 802.15.4g | 802.15.4a/z | NFC |
| Purpose | Low Power Cell Phones | High Bandwidth Ubiquitous | Low power IP-based robust mesh | Low power robust mesh | Location-ing | Contactless |
| Coverage | 10-100m | 30-180m | +/- 30m | 10-100m | <10m | <1m |
| Data Rate | <1Mbps - <100Mbps | <1Mbps - >1Gbps | Typically 250 kbps | 20 Kbps - 250 Kbps | <1Mbps | <1Mbps |
| Example use case | Wearable to smartphone communication | Audio/video transmission. Rasberry-Pi to Cloud/On-Prem communication | Network of many battery-powered devices, such as a smart patient home that require internet connectivity | Network of many battery-powered devices, such as a smart patient home that does NOT require internet connectivity | Wireless Body Area Networks (WBANs), Asset tracking | Medication labeling, Bluetooth/Wi-Fi pairing |

**Figure 4.5:** Connectivity software stack in *User Edge* mapped approximately as OSI layers by NXP (3) and additional characteristics

battery-powered devices, such as a patient's smart home. Thread is IP-based, allowing devices to connect to the internet without a gateway.

*Zigbee* is a low-power and low data rate communication protocol. Embedded devices that are part of the Zigbee network can operate in low power or sleep mode, extending the battery life significantly (68). It is often used for navigation and positioning for short-range purposes. Zigbee devices are non-IP, meaning that a gateway (*Smart Device Edge*) is required to connect to the internet.

*UWB* is nowadays most often used for localization of devices. It does this far more accurately than BLE and Wi-Fi while consuming less energy. Some of its typical use cases are indoor navigation and item tracking. While these do not have a direct use in the domain, the technology plays an important role in creating wireless body area networks (WBANs) (69), typically consisting of sensors placed in, on, or near the surface of the body.

*NFC*, built upon the RFID technology, allows communication between two devices within a very short range. One of the benefits of it being a proximity technology is that it remains inactive until the two devices are close to each other, hence not consuming any power until that is the case. It only requires one of the devices to be powered to open a communication channel. The other device does not need to be powered at all. Due to its range, it has limited use in edge computing systems, but it can be used to accelerate the process of pairing Bluetooth/Wi-Fi devices or exchanging credentials. Another medical use case, less related to edge computing, can be the labelling of medicine with NFC tags, which for example, allows the user to tap the tag with a smartphone to verify whether the medicine should be taken and get extra information about dosages or side effects.

Other short-range technologies not mentioned in fig. 4.5 exist, such as Z-Wave and traditional RFID (68). We mainly focused on the short-range as medical embedded devices tend to be close to the user (patient). However, some use cases might have characteristics comparable to industrial IoT, typically requiring longer ranges. Such technologies can be categorized into two popular forms, cellular networks and low-power wide-area networks (70). Cellular networks include Long-Term Evolution for Machines (LTE-M), Narrowband IoT (NB-IoT), and 5G. The Low-Power Wide-Area networks that can be used are LORAWan, SigFox, Ingenu, Weightless, and DASH7. The cellular networks are, like Wi-Fi, often used to facilitate communication between the *Smart Device Edge* and more resourceful servers such as the *Service Provider Edge* and *Centralized Data Centers* (71).

Independent of what communication technology is used, the vulnerability of edge devices demands a high level of security. Data in transit should be secure, further explained in section 4.2.5. Due to the lack of ports, constrained devices are often updated using over-the-air updates. While doing this securely is mainly ensured by hardware and application decisions, it usually requires a stable connection, and therefore, the communication technology used is relevant.

### 4.2.4 Middleware Layer

This study focuses on the middleware layer that supports the entire spectrum of devices within a healthcare EC system. The middleware should provide interfaces between different system components that would otherwise have difficulty communicating. The discussed middleware platforms are chosen based on their market share in 2021 (63). These include AWS IoT (37%), Microsoft Azure IoT (27%), Google Cloud IoT Platform (22%), IBM Watson IoT platform (15%), Bosch IoT Suite (10%), and Cumulocity (9%).

| **Middleware** | **Communication** | **Security** | **SDK language** | **Marketshare 2021 (63)** |
|---|---|---|---|---|
| AWS IoT | MQTT, HTTP(S) | TLS<br>X.509 CA Certificates<br>IAM | C++, Python, Javascript, Java, Embedded C | 37% |
| Azure IoT | MQTT, HTTP(S), AMQP | TLS<br>X.509 CA Certificates<br>IAM | .NET, Java, Node.js, Python | 27% |
| GCP IoT Platform | MQTT, HTTP(S) | TLS<br>JWT key authentication per device<br>RSA and Elliptic Curve to verify signatures<br>X.509 CA Certificates<br>IAM | Embedded C | 22% |
| IBM Watson IoT Platform | MQTT | TLS<br>X.509 CA Certificates<br>IAM | C++, Python, Java, Node.js | 15% |
| Bosch IoT suite | MQTT, HTTP(S) | TLS<br>X.509 CA Certificates<br>IAM | N/A | 10% |
| Cumulocity | MQTT, SmartREST (on HTTP) | TLS<br>X.509 CA Certificates<br>IAM | C++ | 9% |

**Table 4.4:** Popular edge computing and IoT middleware with their basic characteristics

The design decision for middleware is highly dependent on the cloud vendor used throughout the *Service Provider Edge* and *Centralized Data Center*. Typically it is advised to use the middleware provided by that cloud vendor. From table 4.4 this would result in choosing the products from AWS, Azure, GCP, and IBM. Other cloud-independent middleware could be used to prevent being dependent on a single vendor, such as Bosch IoT suite and Cumulocity. Bosch runs their platform on AWS, but having the infrastructure provider and middleware platform provider being different still reduces the dependency on a single vendor. Cumulocity is built to be able to run on multiple data centers, removing the dependency completely. Typically, the middleware platforms require different middleware to be installed on resource-constrained devices and in the data centers. For example, Cumulocity requires IoT Edge to be installed at the *Constrained/Smart Device Edge* while the *Service Provider Edge* or *Centralized Data Centers* run the IoT Platform.

Regarding the features of the middleware platforms, there is little difference in communication and security methods. However, if developers strongly prefer an SDK programming language, it could be advisable to opt for a certain middleware platform. While Bosch and Cumulocity seem to have few SDK options, all platforms support using direct API calls using REST. Besides the commercial middleware platforms, there are open-source platforms such as EdgeX Foundry[1], KAA[2], and ThingSpeak[3].

### 4.2.5 Data Layer

An important resource within edge computing healthcare systems is the generated data. The sensors in the *Embedded Device Edge* collect the vital parameters of patients, and upon transforming this data into valuable information, healthcare professionals can take appropriate actions. However, dealing with this personal data requires proper security and privacy protection. While guaranteeing this protection, edge computing systems simultaneously have to consider some key performance indicators (KPIs). The large amount of data generated requires a lot of processing before being valuable, and therefore a series of data operations is typically required. Prior research showed that compared to more centralized systems, realizing this protection and achieving acceptable performance on the KPIs is more challenging for edge computing systems (72). Additionally, operating in the healthcare industry comes with extra requirements. Kouicem et al. (73) studied these requirements and additional security challenges. The requirements include authentication, confidentiality, integrity, and privacy. The domain also brings along some challenges, including the limited resources on devices, the required mobility for devices used on human bodies, and the heterogeneity of the different devices used throughout the medical system.

Best practices for data operations within an edge computing system for healthcare were recently studied by Hartman et al. (71). They identified six important operations that have to be considered, which are *transmission and retrieval*, *encryption*, *authentication*, *classification and prediction*, *edge mining*, and *data reduction*. The available and proven techniques are discussed for these operations, except for classification and prediction, which will be further discussed in the section 4.2.6.

[1]https://www.edgexfoundry.org/
[2]https://www.kaaiot.com/
[3]https://thingspeak.com/

#### 4.2.5.1 Transmission and retrieval

*Transmission and retrieval* is an important contributor to the latency of healthcare systems. There are several techniques to realise energy-efficient and low-latency data transmission and retrieval. Sending and retrieving data from edge devices to more centralised computing centers typically increases the latency. Techniques that focus on selecting what data should be sent to a server and what data can be processed closer to the data-producing device are therefore essential. Aazam et al. (74) show that properly dividing the computational work over the different types of edges and the cloud can reduce the overall latency and energy consumption of a system. While the optimal strategy depends on the use case, general optimisation strategies exist for the problem. An example is a strategy proposed by Lu et al. (75) based on deep reinforcement learning and tested using a simulation and Google Cloud.

#### 4.2.5.2 Encryption

*Encryption* comes in a wide variety of techniques. Due to the different energy consumption of these techniques and the differences in resources within edge devices, it highly depends on the situation what technique fits best. Elliptic-curve cryptography (ECC) is typically used within the *User Edge* as it requires only a small key size, which is preferable for devices with limited storage and computation power (3). Table 4.5 provides a variety of other algorithms that can be considered. While ECC is generally the advisable option, in some use cases, such as extreme low-energy use cases, it might be more useful to opt for lightweight primitives. While being more lightweight, they often provide less security and accuracy. The algorithms considered in table 4.5 are well-established algorithms with proper development support. Other, often researched-initiated, algorithms exist. These were extensively studied by Thakor et al. (76). Their study compared 41 lightweight cryptography algorithms for resource-constrained devices over seven performance metrics. These performance metrics are based on a report on resource-constrained IoT devices by the National Institute of Technology (NIST) (77).

| **Encryption technique** | **Use case** | **Example reference** |
|---|---|---|
| Elliptic curve cryptography (ECC) | Low-latency encryption | Diro et al. (78) |
| Modified elliptic curve cryptography (MECC) | Low-latency and low-energy encryption | Chanti et al. (79) |
| Advances Encryption Standard (AES) | Most common symmetric key algorithm | Yumnam et al. (80) |
| Tripple DES (3DES) | Secured by three different keys, requires more memory | Santa et al. (81) |
| Rivest-Shamir-Adleman (RSA) | Efficient in verifying signature and encrypting | Suarez et al. (82) |
| Digital Signature Algorithm (DSA) | Compliance with US federal government, efficient in verifying signature and encrypting | Anusha et al. (83) |
| Blowfish | Low-latency, low-energy encryption | Bhalaji et al. (84) |
| Tiny Encryption Algorithm (TEA) | Low-latency, low-energy encryption | Rajesh et al. (85) |

**Table 4.5:** Encryption algorithms

#### 4.2.5.3 Authentication

*Authentication* is closely related to encryption and is important to ensure privacy in healthcare systems. It refers to the process of verifying an identity. When using traditional clouds or *Service Edges* there are typically two types of authentication: client and server. From the client side, embedded devices can authenticate themselves by providing a certificate (often X.509 client certificate), while smart devices and other more resourceful devices can use more advanced Identity and Access Management solutions provided by the cloud. From the server's side, when devices attempt to reach the server, it has to provide those devices with a certificate of its own (often X.509 certificate chain over TLS). While in practice, the cloud-provided solutions are used, there is a considerable amount of research (86) (87) (88) being conducted on blockchain-enabled access management for edge computing systems. The results show promising features of such approaches, with better data security guarantees for authentication, auditability, and confidentiality.

#### 4.2.5.4 Edge mining and reduction

*Edge mining and data reduction* aim to decrease the amount of data transmitted between different types of edges and primarily focus on limiting the amount of traffic between the embedded devices and the centralized cloud. By doing a part of the data processing at the *Embedded* or *Smart Device Edge*, the number of packets that need to be sent to the cloud can be reduced. There are several techniques to achieve this goal, from (lossless) compressing the data and decompressing it in the cloud (89), to doing a part of the analysis at the edge, for example, by detecting abnormal values on the embedded device itself and only sending those values to the cloud (90).

#### 4.2.5.5 Security

An emerging technology in data security is federated learning, a machine learning technique that enables multiple decentralised devices holding local data to train a model together without exchanging the data. The local devices only send their local model updates to a central server, which allows them to collaboratively create a global model which is, in return, distributed. As the medical domain often deals with sensitive data, this is a promising approach to keep the data at the edge without being stored in a cloud. In this way, if the cloud's security is breached, no personal data is being compromised. Several related works (91) (92) (93) (94) have shown the practical implementations of federated learning in the domain of healthcare. From an industry point of view, large cloud vendors provide instructions on how federated learning can be realised using their infrastructure [1] [2].

Regarding storage options, for the *On-Premise Data Centers*, *Service Provider Edges*, and *Centralized Clouds* it highly depends on the type of data that has to be stored which option is advisable. Typical choices are *object storage*, *file systems*, and *block storage.*

### 4.2.6 Application Layer

To create a system that adds value to the medical state of users, the Application layer uses all other layers to create the end products used in the medical domain. There are many possible applications in the medical domain, such as identity recognition, biometric identification, real-time monitoring of vital signs, and telemedicine (95). Due to the heterogeneity of healthcare applications, there is no single stack of application features relevant to all use cases. However, some features are most commonly used and supported by cloud vendors throughout all the layers. These include *real-time analytics, machine learning and artificial intelligence, analytics, visualization, event and reporting, authentication* and *logging and monitoring* (96) (97). The three largest cloud vendors (AWS, GCP, AZURE) support these features, and additionally, their support was researched by Agarwal and Alam (96). Their study shows in more detail the support that different clouds vendors provide for hardware, middleware, and applications.

[1]https://aws.amazon.com/blogs/architecture/applying-federated-learning-for-ml-at-the-edge/
[2]https://cloud.google.com/architecture/federated-learning-google-cloud

#### 4.2.6.1 Security

The security of a medical edge computing system is highly dependent on decisions made in the application layer. Since other layers also have a large impact, we provide a single overview of security threats in this section for the sake of clarity. The importance of a well-secured system was clearly emphasised in 2021 when over 61 million entries of data originating from wearable devices were exposed online[1]. Each layer within an edge computing system brings different challenges, table 4.6 shows important security threats in edge computing systems and possible countermeasures. It was based on three scientific publications (5) (7) (6), a book on state-of-the-art edge computing methods (3), and the security best practices of AWS[2], GCP[3], and Azure[4]. Sources to find more information about a given countermeasure or threat are provided and should help developers find the relevant resources swiftly. While this study focused on the most common threats and challenges, there are endlessly more existing, especially regarding the constrained devices. We refer to the Baseline Security Recommendations for IoT published by the European Union Agency for Network and Information Security (98) for a more extensive deep dive into this topic.

[1]https://www.zdnet.com/article/over-60-million-records-exposed-in-wearable-fitness-tracking-data-breach-via-unsecured-database/

[2]https://docs.aws.amazon.com/iot/latest/developerguide/security-best-practices.html

[3]https://cloud.google.com/iot/docs/concepts/device-security

[4]https://docs.microsoft.com/en-us/azure/iot-fundamentals/security-recommendations

| Layer | Threat/Challenge | Description | Counter measure | Source(s) |
|---|---|---|---|---|
| Connectivity | DoS / DDoS | Floading of server by sending overwhelming amount of messages to network | Detect D(D)oS with ML, bandwidth control, load-balancing | (7) (5) |
| | Insecure nearest node discovery | By using Neighbor Discovery Protocol can find MAC adressess of available routers, can lead to D(D)oS | ECC/RSA signatures | (6), GCP |
| | Buffer overflow | Sending incomplete packets to overflow buffer | Make use of threat hunting modules | (6) |
| | Session hijacking | Generating fake messages to realise DoS | Secret keys, lightweight encryption systems | (6) |
| | RPL routing attack | Vulnerable Low Power and Lossy Networks, Sink/Worm-hole attacks, tamper data and cause delays. | Hashing, IDS, encrypted keys, signal strength monitoring, message integrity | (6) NXP(3) |
| | Replay attack | Malicious devices sending duplicated fragmented packets | Timestamps, authentication for packets verification, IDS | (6), (7) |
| | Sybil attack | Single malicious device using multiple identities | Access control list, analysis on interaction | (6), (7) |
| | Frequency jamming | Creating noise signals in a transmission frequency | Measuring signal strength, packet delivery ratio, encoding packets, changing frequencies/location | (6), (7) |
| | Spoofing | Unauthorized packets sent into the network | Measuring signal strength, encode data in transit | (6) (7) |
| Hardware | Default passwords | Constrained edge devices that are out-of-the-box often have default passwords | Change default passwords | NXP(3) |
| | Purpose-built chipsets | Custom chipsets tailored for specific use-cases are often not tested extensively and vulnerabilities might be found | Hardware monitoring | (6) |
| | Insecure interfaces | Insecure interfaces to the real-world | Block or restrict external interfaces and ports | (6) |
| | Secured environments | The need to run specific applications such as booting and firmware updates in secured environments | Secured enclaves, trusted execution environments | NXP(3) |

35

| Layer | Threat/Challenge | Description | Counter measure | Source(s) |
|---|---|---|---|---|
| Application | Browser based | Unwanted software by infected websites | ML frameworks to detect malicious activity | (7) |
| | Phishing | Fraudulant messages to steal sensitive information | Anti-phishing techniques (ML) and education | (7) |
| | MQTT/HTTP/CoAP attack | Target network protocols to reduce data transfer performance | Use proper security (DTLS, TLS, SSL) | (6), AWS, GCP, Azure |
| | Code injection | Injection of code by processing invalid data | Input validation | (5) |
| | Insecure interfaces | Exploitation of insecure APIs and other interfaces | Password strength, secure coding, firewalls | (6) |
| | Elevation of privilege | Users with limited permission enhance their permission without authorization | Principle of least privilage | (5), AWS, |
| | Privacy violation in cloud | Service providers might violate data privacy | SLAs and policies | (6) |
| Data | Data at rest | Data stored on constrained edge devices must be encrypted | dm-crypt, BitLocker, prince algorithm for on-the-fly automatic encryption | NXP(3) |
| | Data in transit | Data in transit from or towards constrained edge devices must be encrypted | IPsec, TLS, SSH, TMP, I2C interface | NXP(3) |
| Middleware | Middleware security | Communication between edge devices and the cloud must be secure | IAM, logging and monitoring, secure device communication, X.509 certificates | AWS, GCP, Azure |
| Operating system | Insecure firmware / OS | Can be misconfigured and host malicious applications | Regularly update, root of trust, secure boot | (6) |

**Table 4.6:** Security threats and challenges based on (5) (6) (7) (3) and AWS, GCP, Azure

#### 4.2.6.2 Regulation

Regulation plays an important role in the design of healthcare systems. Due to the difference in laws and regulations between countries, it is impossible to create a single framework that captures all necessary considerations. In this study, we have focused on outlining the relevant laws and regulations for healthcare applications deployed in The Netherlands. The motivation behind this decision was that the research was conducted in The Netherlands. The laws and regulation in the domain of Dutch healthcare can be divided into four categories: *regulatory legislation*, *intellectual property (IP) rights*, *law of obligations*, and *EU-related laws and legislation.* Table 4.7 shows the vast amount of documents that must be considered when developing a healthcare system in The Netherlands. While large cloud vendors often provide information on which of their services comply with well-known regulations, such as HIPPA and the GDPR, the country-specific legislation is not mentioned. These should be thoroughly studied before developing a healthcare system. In the case of the European Union, legislation on medical devices is often changed and updates can be tracked on the official European Medicines Agency website[1]. Country-specific regulation should be investigated by organisations developing EC healthcare systems, as now generic guidelines exist to comply with it. However, overviews of relevant laws and legislation, such as table 4.7, provide easy access to relevant work and can guide the organisations to achieve compliance.

For other regulations, such as those on the European level, standards exist. ISO 13485[2] is an international standard that many manufacturers of medical equipment adhere to, it ensures compliance with both MDR and IVDR. A combination of ISO 27001[3] and 27701[4] compliance allows developers to cover most aspects of the GDPR.

[1]https://www.ema.europa.eu/en/human-regulatory/overview/medical-devices: :text=The%20Medical%20Devices%20Re

[2]https://www.iso.org/standard/59752.html

[3]https://www.iso.org/isoiec-27001-information-security.html

[4]https://www.iso.org/standard/71670.html

| Regulatory legislation | IP rights | Law of obligations | EU laws and regulation |
|---|---|---|---|
| Gedragscode Gezondheidsonderzoek *Code of Conduct health research* | Auteurswet *Copyright law* | Productaansprakelijkheid *Product liability* | General Data Protection Regulation |
| Wet kwaliteit, klachten, en geschillen in de zorg *Healthcare Quality, Complaints and Disputes Act* | Octrooiwet *Patent law* | Beroepsgeheim *Professional confidentiality* | European Medical Device Regulation (EU-MDR) |
| | Databankwet *Database law* | Geneeskundige behandelingsovereenkomst *Medical treatment agreement* | European In-Vitro Diagnostic Devices (EU IVDR) |
| | | | European Pharmaceutical Legislation |

**Table 4.7:** Relevant laws and legislation for healthcare systems deployed in The Netherlands

#### 4.2.6.3 Evaluation

Evaluation metrics have become difficult to set for edge computing systems due to the wide variety of applications. Even if generic evaluations are performed, it has been pointed out by Xiao et al. (99) that this is not always sufficient. Their study focused on video analytics at the edge and concluded that evaluations in existing literature are often incomplete and lead to premature conclusions or unclear results. In the case of video analytics, it was shown that the characteristics of video content are very influential on the accuracy achieved by analytical pipelines. To this end, they proposed YODA, a benchmark capable of clarifying and predicting the performance of video analytical pipelines. Developers should aim to use such unbiased frameworks for all relevant performance evaluation metric

To create a set of performance evaluation metrics for healthcare edge computing systems, several scientific papers were studied. It must be mentioned that not all papers are from the medical domain due to the lack of publications. Table 4.9 shows the possible evaluation metrics and in which related work it was used. The related works are focused on applications or simulations of edge computing systems. Accuracy and latency are the most popular evaluation metric, while other metrics such as cost, ease of deployment, fault tolerance, and device lifetime are barely used. This can partly be explained by the fact that many studies focus on a specific goal, not creating a commercial system. However, some of the metrics that are barely used in related works are especially relevant to the medical domain, such as reliability, availability, and fault tolerance. Research efforts have focused primarily on making the systems as fast and accurate as possible, while for medical applications, these might not be the most important metrics.

To simulate the performance of an edge computing system, frameworks and simulators can provide aid. While the commercial cloud vendors typically focus on specific parts of an edge computing system in their simulation tools, research has focused on creating more complete simulations. Table 4.8 shows the simulators and frameworks.

Overall, it is recommended to use simulations and unbiased evaluation frameworks before developing a medical edge computing system.

| **Study or product** | **Year** | **Description** |
|---|---|---|
| Daga et al. (100) | 2019 | Castnet: an evaluation framework for MEC |
| Jha et al. (101) | 2019 | IoTSim-Edge: simulate heterogeneous IoT and edge computing infrastructure |
| Hasenburg et al. (102) | 2021 | Mockfog2: testing full edge computing applications on simulated infrastructure |
| AWS IoT Device Simulator | Ongoing | Test device integration and IoT backend services without using physical devices |
| Azure IoT Device Telemetry Simulator | Ongoing | Simulate telemetry data to Azure products without using physical devices |

**Table 4.8:** Simulation tools available

| Study Metric | | | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 2019 | 2020 | | | | | | | | | | | 2021 | | | | | | | | | | | | 2022 | |
| | (100) | (103) | (104) | (105) | (106) | (107) | (108) | (109) | (110) | (111) | (112) | (113) | (102) | (114) | (115) | (116) | (117) | (118) | (119) | (120) | (121) | (122) | (123) | (124) | (125) | (126) | (127) |
| Resource utilization | | | | ✓ | ✓ | | | | | ✓ | | ✓ | | | | ✓ | | ✓ | | | ✓ | | ✓ | | | ✓ | |
| Resource load | | | | ✓ | ✓ | | | | | | | | | | | | | ✓ | | | ✓ | | ✓ | | | ✓ | |
| Accuracy | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | | | | | | ✓ | | ✓ | ✓ | ✓ | | | | ✓ | ✓ | | |
| Device lifetime | | | | | | | | | | | | | | | | | | | | | | | | | | ✓ | |
| Response time | | | ✓ | | | | | | | ✓ | | ✓ | | ✓ | | | ✓ | | | ✓ | | ✓ | ✓ | | ✓ | | |
| Latency time | | | ✓ | ✓ | ✓ | | | ✓ | | ✓ | | | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | | | ✓ | | ✓ | ✓ | |
| Throughput | | | ✓ | | | | | | | | | | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | | ✓ | | | | |
| Scalability | | ✓ | | | | | | | | | | | | ✓ | | | ✓ | | | | | | | | | | |
| Overhead | | | | | | | | | | | | | | | ✓ | ✓ | ✓ | | | | ✓ | | | | | | |
| Energy consumption | ✓ | | | | | | ✓ | | ✓ | ✓ | | ✓ | | | | | ✓ | | | | | ✓ | | | | | ✓ |
| Reliability | | | ✓ | | | | | | | ✓ | | | | | | | | | | | | | | | | | |
| Availability | | | | | | | | | | ✓ | | | | | | | | | | | | | | | | | |
| Security | ✓ | | | | | ✓ | ✓ | | ✓ | | ✓ | | | | | | | | | ✓ | ✓ | ✓ | | ✓ | | | |
| Privacy | ✓ | | | | | | ✓ | | | ✓ | | ✓ | | | | | | | | ✓ | ✓ | ✓ | | ✓ | | | |
| Fault tolerance | | | | | | | | | | | | ✓ | | | | | | | | | | | | | | | |
| Cost | | ✓ | | | | | | | | | | | | | | | | | | | | ✓ | | | | ✓ | |
| Ease of deployment | | | | | | | | | | | | | | | | | | | | | | | | | | | |

**Table 4.9:** Performance evaluation metric for edge computing systems

## 4.3 Reference architecture

The Reference Architecture Healthcare Edge Computing (RAHEC) is shown in fig. 4.6. It provides a high-level overview of all components in an edge computing system for healthcare and how different components can interact. The two main components are the *User Edge* and the *Service Provider Edge and Centralized Data Center*, which are further split up into the layers from the Reference Model (fig. 4.2, originating from the LF-Edge taxonomy fig. 2.1). The reference architecture was made using an Amazon Web Services stack. However, due to the similarity in product offerings between different cloud vendors, the architecture can easily be converted to another cloud vendor. The involvement of different possible stakeholders is represented in the reference architecture by showing with which components they can interact.

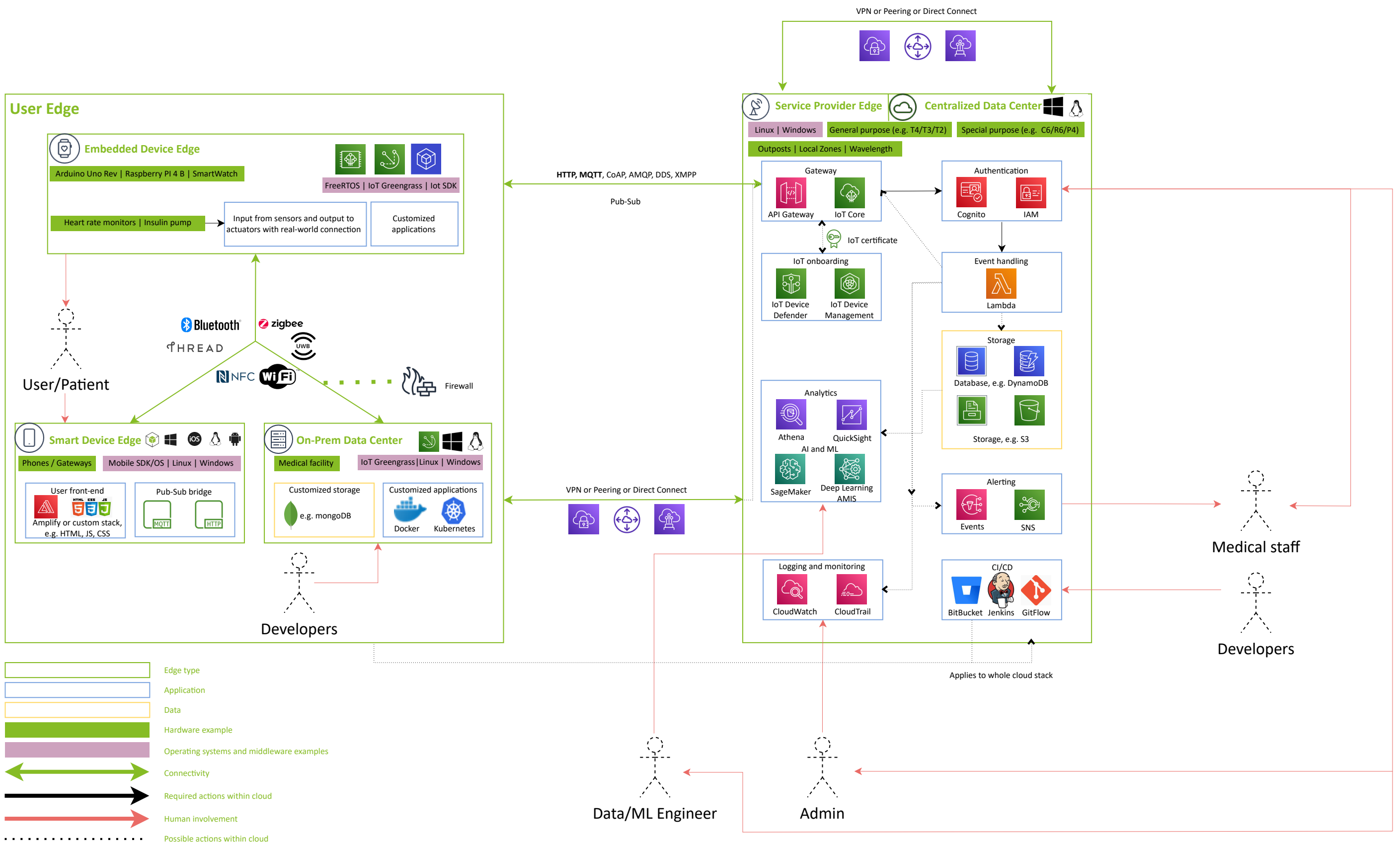
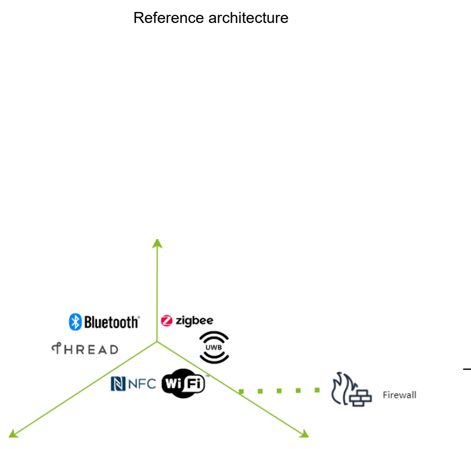
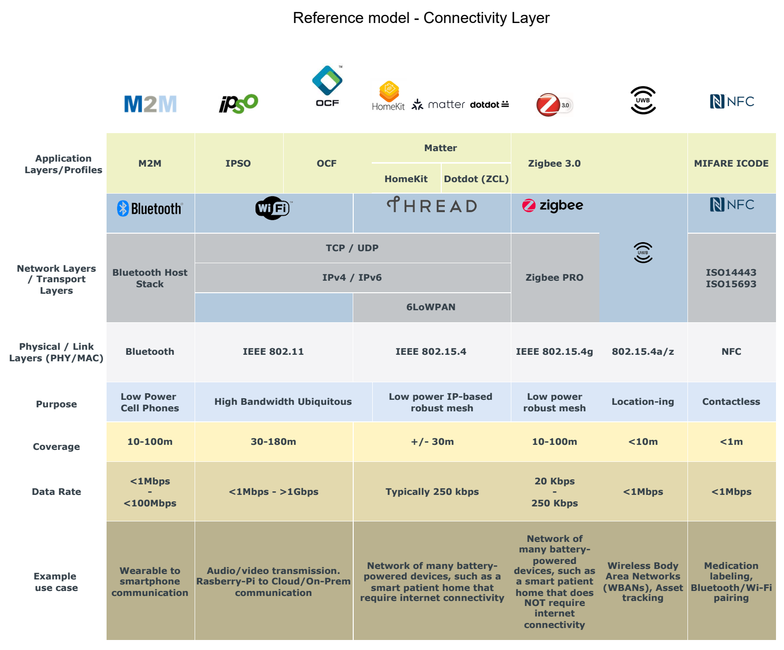


**Figure 4.6:** Reference Architecture Healthcare Edge Computing

While the reference architecture provides a quick overview of the recommended structures in an edge computing healthcare system, there is a lack of detailed information that developers can use to develop a system. For that purpose, the detailed information on each layer given in section 4.2 can be used. By combining the artifacts, the reference architecture functions as a map to find relevant information on recommended products, services, and technologies for a specific component, leading to faster design decisions.

This methodology of finding relevant information using both artifacts is illustrated in fig. 4.7, where the focus in the reference architecture is on the connectivity between the *Embedded Device Edge*, *Smart Device Edge*, and *On-Prem Data Center*. The reference architecture provides a brief overview of the relevant technologies that *could* be used. Complementing this with details on the features of each of the technologies allows developers to decide which technology *should* in their use case.




Reference model - Connectivity Layer
M2M
IPSO
OCF
HomeKit
matter
dotdot
3.0
UWB
NFC

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Application Layers/Profiles | M2M | IPSO | OCF | Matter / HomeKit | Matter / Dotdot (ZCL) | Zigbee 3.0 | | MIFARE ICODE |
| | Bluetooth | WiFi | | THREAD | | zigbee | UWB | NFC |
| Network Layers / Transport Layers | Bluetooth Host Stack | TCP / UDP<br>IPv4 / IPv6 | TCP / UDP<br>IPv4 / IPv6<br>6LoWPAN | | | Zigbee PRO | UWB | ISO14443<br>ISO15693 |
| Physical / Link Layers (PHY/MAC) | Bluetooth | IEEE 802.11 | IEEE 802.15.4 | | | IEEE 802.15.4g | 802.15.4a/z | NFC |
| Purpose | Low Power Cell Phones | High Bandwidth Ubiquitous | | Low power IP-based robust mesh | | Low power robust mesh | Location-ing | Contactless |
| Coverage | 10-100m | 30-180m | +/- 30m | | | 10-100m | <10m | <1m |
| Data Rate | <1Mbps - <100Mbps | <1Mbps - >1Gbps | Typically 250 kbps | | | 20 Kbps - 250 Kbps | <1Mbps | <1Mbps |
| Example use case | Wearable to smartphone communication | Audio/video transmission. Rasberry-Pi to Cloud/On-Prem communication | Network of many battery-powered devices, such as a smart patient home that require internet connectivity | | | Network of many battery-powered devices, such as a smart patient home that does NOT require internet connectivity | Wireless Body Area Networks (WBANs), Asset tracking | Medication labeling, Bluetooth/Wi-Fi pairing |

**Figure 4.7:** Reference architecture subset complemented by information of the Connectivity layer section 4.2.3

# 5

# Proof of Concept

In this section, we discuss our proof of concept. First, the artificial medical scenario is explained. Next, the technical set-up is explained and the influence of the reference architecture and model are discussed. Lastly, the shortcomings revealed by the proof of concept are discussed.

We have considered a medical scenario where a patient in a wheelchair wants to inform medical respondents in the event of a fall over. To realise this, the patient has a *constrained device* implanted on the wheelchair, which communicates with a *smart device edge* in case of a fall over. This *smart device edge* in turn communicates with the cloud. In the cloud, the nearest medical respondent is found and alerted on their smartphone. Only when the patient in the wheelchair is standing upright again will a notification be sent via the same route (embedded edge → smart edge → cloud → smartphone) to indicate that the problem has been solved. The structure of using these three types of computing options was chosen as it is the most used in identified related works (Scenario 2 from fig. 4.1).

The Proof-of-Concept for this medical scenario is a simplified solution that proves the feasibility of creating a real-world product using the proposed reference architecture. Figure 5.1 shows the set up, and what components represent the *wheelchair*, *patient-smartphone*, *medical-cloud*, and *respondent-smartphone.* The wheelchair consists of an Arduino Uno R3 board, which is a constrained device edge. The technical specifications are given in table 5.1[1]. The patient's smartphone is represented by an HP-Elitebook x360 1040 G6. The chosen cloud is AWS, motivated by the fact that it has the largest market share of all public cloud vendors (27). The respondent's smartphone can be represented by any device capable of accessing an email client. The different trigger events and actions of the proof-of-concept are shown in fig. 5.2. Details are discussed for each component.

[1]https://docs.arduino.cc/hardware/uno-rev3

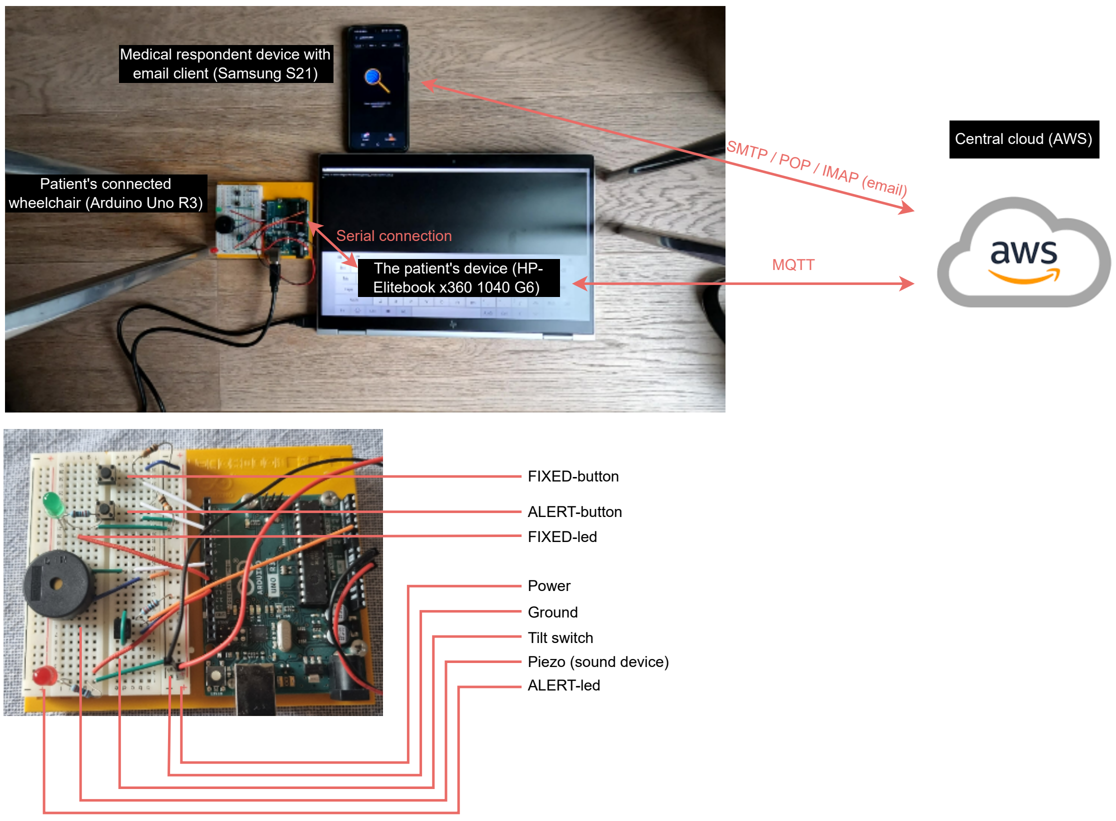

Medical respondent device with email client (Samsung S21)
Central cloud (AWS)
SMTP / POP / IMAP (email)
Patient's connected wheelchair (Arduino Uno R3)
Serial connection
MQTT
The patient's device (HP-Elitebook x360 1040 G6)
aws
FIXED-button
ALERT-button
FIXED-led
Power
Ground
Tilt switch
Piezo (sound device)
ALERT-led

**Figure 5.1:** Proof of concept set-up with embedded device explanation

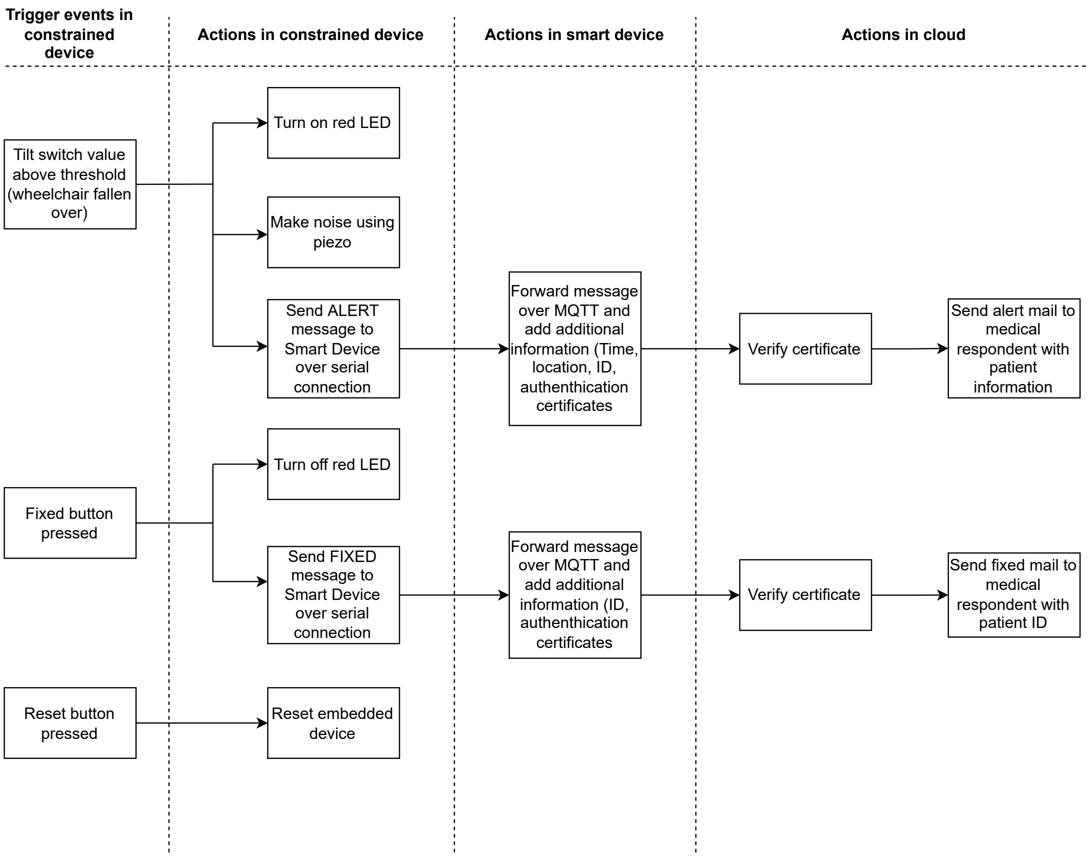

Trigger events in constrained device
Actions in constrained device
Actions in smart device
Actions in cloud
Tilt switch value above threshold (wheelchair fallen over)
Turn on red LED
Make noise using piezo
Send ALERT message to Smart Device over serial connection
Forward message over MQTT and add additional information (Time, location, ID, authenthication certificates
Verify certificate
Send alert mail to medical respondent with patient information
Fixed button pressed
Turn off red LED
Send FIXED message to Smart Device over serial connection
Forward message over MQTT and add additional information (ID, authenthication certificates
Verify certificate
Send fixed mail to medical respondent with patient ID
Reset button pressed
Reset embedded device

**Figure 5.2:** Connected wheelchair PoC trigger events and actions

**Constrained Device Edge (Arduino Uno R3)** has a tilt switch that detects whether it has fallen over. In the event of a fall-over, a red LED and piezo (sound device) alert people nearby that the wheelchair has fallen over. These two signals are used as 'independent' alerts, meaning that even if the edge device has no connection to the smart edge or cloud, it can perform simple tasks independently. Additionally to the independent alerts, in case of a fall over, the *Constrained Device Edge* sends an "ALERT"-message to the *Smart Device* over a serial connection. Furthermore, it has a button "FIXED"-button to indicate that the problem has been solved. If this button is pressed, the red LED and piezo are turned off, and a "FIXED"-message is sent over the serial connection. Lastly, a "RESET"-button and green LED are present on the wheelchair. If the "RESET"-button is pressed, the wheelchair is reset, and the green LED will blink to indicate that the reset is successful. The logic of this device is written in FreeRTOS, of which the programming language is C++.

| | | |
|---|---|---|
| **Pins** | Built-in LED Pin | 13 |
| | Digital I/O Pins | 14 |
| | Analog input pins | 6 |
| | PWM pins | 6 |
| **Communication** | UART | Yes |
| | I2C | Yes |
| | SPI | Yes |
| **Power** | I/O Voltage | 5V |
| | Input voltage (nominal) | 7-12V |
| | DC Current per I/O Pin | 20 mA |
| | Power Supply Connector | Barrel Plug |
| **Clock speed** | Main Processor | ATmega328P 16 MHz |
| | USB-Serial Processor | ATmega16U2 16 MHz |
| **Memory** | ATmega328P | 2KB SRAM, 32KB FLASH, 1KB EEPROM |
| | Weight | 25 g |
| **Dimensions** | Width | 53.4 mm |
| | Length | 68.6 mm |

**Table 5.1:** Arduino UNO R3 technical specifications

**Smart Device Edge (HP-Elitebook x360 1040 G6)** functions as a gateway between the edge and the cloud. To communicate with the *constrained edge device* it listens to a serial port and waits for messages to be sent. As soon as a message is received, it adds additional information and forwards it to the cloud. The patient's ID, location, required aid, and time of the accident are generated by the *smart device.* It is also responsible for the authentication to the cloud, which is done by providing the correct X.509 certificates generated by AWS's Identity and Access Management (IAM) service. Once the patient data is generated and the proper authentication is handled, it publishes the message to an MQTT topic. The logic of this device is programmed in python running on a Linux OS, using the AWS IoT SDK and other python-libraries (py-serial and python-requests).

**Cloud (AWS)** is subscribed to the MQTT topic and consumes the messages sent by the *smart device.* It verifies the certificates, and in case they are valid, the AWS IoT Core sends the data to Lambda (a serverless function service). The Lambda function uses another service, Simple Email Service, to send an email to the medical respondent. In case the MQTT message was an "ALERT", the medical respondent receives the patient data. In the case of a "FIXED"-message, the medical respondent is informed that the aid is no longer required. The performance of different AWS services and possible errors can be tracked using CloudWatch and CloudTrail.

The influence of the reference model and architecture is visible in several design decisions. The structure of using an *constrained device* - *smart device* - *cloud* fits in to the recommended structure of dividing the *User Edge* and *Centralized Data Center* into separate components. Regarding *hardware*, the *Constrained and Smart devices* adhere to the recommended specifications, and the hardware decisions in the cloud are irrelevant as we were not using Infrastructure-as-a-Service, only Platform- and Software-as-a-Service. The *operating systems* chosen, FreeRTOS and Linux, adhere to the recommendations. FreeRTOS was chosen because it has a small footprint, is supported by the AWS-stack, allows multiple tasks to run simultaneously, and has convenient libraries for serial communication. Linux, running on the *smart device*, is the typical choice for gateway devices. In case the proof-of-concept was implemented using a smartphone, Android and iOS would have been more logical choices. *Connectivity* of the proof-of-concept consists of serial connection between the *constrained* and *smart device* and a MQTT-exchange between the *smart device* and *cloud.* The serial-USB connection was chosen due to the lack of network modules for the Arduino Uno R3, more logical would be a Bluetooth-low-energy connection using

MQTT. The MQTT exchange is the recommended protocol for AWS IoT Core, additionally, section 4.2.3 discussed several reasons (latency, bandwidth, and energy consumption) why MQTT is the preferred protocol. The *middleware layer* consists of FreeRTOS at the *constrained device* and AWS IoT Core in the *smart device* and *cloud*. Using this middleware stack allowed the usage of MQTT, X.509 CA Certificates and the support for Python in the SDK. It is in line with the recommendations, which stated that it is typical to use the middleware provided by the used cloud vendor. The *data layer* should ensure data-security at-rest and in-transit. In the PoC, the only data that is stored (at-rest) is on the gateway device, and it solely includes a patient's ID, no personal information, which makes the storage easier because no regulation has to be considered. No patient data is stored in the cloud, only processed on the fly. Regarding data in transit, the serial connection between the *constrained* and *smart* device can be seen as insufficient because there is no encryption, however, as there is a physical connection, it is assumed the user of the PoC can ensure that no security breaches occur. As mentioned before, using a Bluetooth-low-energy connection would be preferable, and in such case, a secure MQTT connection could be used between the *constrained device* and an AWS IoT MQTT broker on the *smart device*. The MQTT connection between the *patient* and *cloud* is secured using TLS 1.2 and the requirement of a verified X.509 certificate. The *application layer* provided information on application features that could be used for specific use cases. This was heavily used during the development of the PoC (for example, by using the dedicated Simple Email Service). Regarding the overall security of the PoC, not all threats/challenges mentioned in table 4.6 were considered and mitigated. Some of these were not considered as they were irrelevant, such as the *frequency jamming* that has no impact on the *constrained device* as it has no dependence on frequencies, and others, such as *data at rest*, should be covered if the connected-wheelchair was to be implemented as a real product.

Besides showing the validity of the reference model and architecture, the goal of the proof-of-concept was to reveal the shortcoming of this study. First of all, reoccurring advice is to follow the recommendations of the cloud provider, this however led to a large dependence on AWS in the proof-of-concept (RTOS, middleware, and connectivity) which causes the overall system to be less flexible when it comes to shifting over to another cloud vendor. By using more platform-independent products/technologies, this could be prevented, for example, by replacing AWS IoT with Cumulocity. Next, for the cross-layer concerns *real-time*, *scalability*, and *resilience* basic information on which products and technologies can be used to ensure these are covered in a specific layer is provided. However, no recommendations are given on ensuring coverage throughout the system. An example is the usage of freeRTOS on the *constrained devices*, which allows for *hard* real-time inferences and resilience, but without a *smart device* that is capable of ensuring the same real-timeness and resilience, the complete healthcare system is still unable to guarantee solutions to the cross-layer concerns. Lastly, the reference architecture (fig. 4.6) shows an overwhelming amount of application features, of which typically only a small subset will be used per use-case, causing the main goal of making design decisions easier to be partly compromised.

A complete replication package, including detailed installation instructions and a video showing the working of this PoC, is provided on the GitHub page of this proof-of-concept[1].

[1]https://github.com/lucasdegeus/master_thesis_ldg_PoC

# 6

# Conclusion and Future Work

We have introduced the fast-growing field of edge computing for healthcare as a solution to overcome the shortcomings of traditional cloud computing. The field has gained significant attention from industry and academia in the form of research, investments, and available products. This creates new capabilities and opportunities for edge computing healthcare systems. However, it requires more and more design decisions to be considered during the development of such a system. While reference models and architectures for similar fields exist, the healthcare domain brings additional requirements such as higher availability, response time, robustness, and compliance with regulations.

To support developers in the medical domain, we studied how an edge computing system for healthcare should be architected (RQ). Taking a similar approach as the authors of RAMI4.0 and RAMEC, we have created a three-dimensional (6x6x6) reference model consisting of layers, types of edges, and cross-layer concerns. This results in 216 different views that could be considered during the development of a healthcare edge computing system. Detailed information on recommended products and structures for each layer is given, allowing developers to make faster design decisions. To support this, we proposed the Reference Architecture Healthcare Edge Computing (RAHEC), which provides a high-level overview of the recommendations that originate from the reference model. It can be used as a map to find relevant information on products, services, and technologies for specific components. The architecture uses AWS products and services but can easily be converted to other popular clouds as the information per layer considers multiple clouds.

To evaluate the usefulness and possible shortcomings of RAHEC, we implemented a proof-of-concept. The PoC represents a connected wheelchair that can alert humans in the direct physical environment of the patient and a medical respondent via email in case of a fall over. The PoC used a constrained device, a gateway, and a cloud. It was shown that RAHEC could be used for many design decisions throughout the different layers. The main issues of the current RAHEC that were revealed by the PoC include the bias towards cloud vendor products/services, the focus on solving cross-layer concerns per layer instead of throughout the entire system, and a vast amount of features shown in the architecture while only a small subset is used per use-case.

Additionally, we studied how developers can comply with regulations (SQ1) and evaluate their medical systems (SQ2). We used The Netherlands as a case study to identify all relevant laws and legislation for medical applications and provided international standards that help developers achieve compliance. When it comes to evaluation, we found that it is highly dependent on the use case, and unbiased frameworks should be used. We provided 17 possible evaluation metrics and referenced related work for each metric.

The proposed reference model and RAHEC form a basis for standardising the development of healthcare edge computing systems. However, the acceptance and adoption of the proposed artifacts are most important for them to become valuable. Therefore, future work should focus on identifying other state-of-the-art works and extending the proposed artifacts with these works. As the field is very industry-driven, it is important to gain support from large organisations. The cross-layer concerns should be emphasised more in future work to ensure that systems developed using RAHEC are functional throughout all layers. The development of more comprehensive proofs-of-concept and possibly real-world applications will bring even more shortcomings of RAHEC to light. From the knowledge of these shortcomings, the models can be further improved. Lastly, the information provided per layer must be kept up to date to ensure that state-of-the-art products, services, and technologies are recommended. This can be realised by closely following new developments provided by academia and industry.

# References

[1] Linux Foundation Edge. **State of the Edge 2021: A Market and Ecosystem Report for Edge Computing**, 2021. v, 6

[2] Deutsches Institut fr Normung. **Referenzarchitekturmodell Industrie 4.0 (RAMI4.0)**, 2016. v, 9, 10

[3] Robert Oshana. *NXP: Essentials of Edge Computing*. BookBaby, 1 edition, 2022. v, vii, 19, 23, 26, 27, 31, 34, 35, 36

[4] Jasenka Dizdarević, Francisco Carpio, Admela Jukan, and Xavi Masip-Bruin. **A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration**. *ACM Comput. Surv.*, **51**(6), jan 2019. vii, 25

[5] Tzu Wei Tseng, Chia Tung Wu, and Feipei Lai. **Threat Analysis for Wearable Health Devices and Environment Monitoring Internet of Things Integration System**. *IEEE Access*, **7**:144983–144994, 2019. vii, 34, 35, 36

[6] Hamed HaddadPajouh, Ali Dehghantanha, Reza M Parizi, Mohammed Aledhari, and Hadis Karimipour. **A survey on internet of things security: Requirements, challenges, and solutions**. *Internet of Things*, **14**:100129, 2021. vii, 34, 35, 36

[7] Shakila Zaman, Khaled Alhazmi, Mohammed A Aseeri, Muhammad Raisuddin Ahmed, Risala Tasin Khan, M Shamim Kaiser, and Mufti Mahmud. **Security threats and artificial intelligence based countermeasures for internet of things networks: a comprehensive survey**. *Ieee Access*, **9**:94668–94690, 2021. vii, 34, 35, 36

[8] Stephanie B Baker, Wei Xiang, and Ian Atkinson. **Internet of things for smart healthcare: Technologies, challenges, and opportunities**. *Ieee Access*, **5**:26521–26544, 2017. 1

[9] Ayten Ozge Akmandor and Niraj K Jha. **Smart health care: An edge-side computing perspective**. *IEEE Consumer Electronics Magazine*, **7**(1):29–37, 2017. 1

[10] Kevin Westcott Ariane Bucaille, Gillian Crossan. **Deloitte's Technology, Media, and Telecommunications predictions**. pages 64–71, 2022. 1

[11] Beth Musumeci, Ralph Ramsey, and Stephan Brennan. **Medical devices are vital, but vulnerable: Treat infrastructure risks to safeguard patient care**, 2020. 1

[12] Clemens Scott Kruse, Rishi Goswamy, Yesha Jayendrakumar Raval, and Sarah Marawi. **Challenges and opportunities of big data in health care: a systematic review**. *JMIR medical informatics*, **4**(4):e5359, 2016. 1

[13] Alvin Afuang. **IDC: IoT Growth Demands Rethink of Long-Term Storage Strategies**, 2020. 1

[14] Iman Azimi, Arman Anzanpour, Amir M Rahmani, Tapio Pahikkala, Marco Levorato, Pasi Liljeberg, and Nikil Dutt. **HiCH: Hierarchical fog-assisted computing architecture for healthcare IoT**. *ACM Transactions on Embedded Computing Systems (TECS)*, **16**(5s):1–20, 2017. 1

[15] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. **Edge computing: Vision and challenges**. *IEEE internet of things journal*, **3**(5):637–646, 2016. 1

[16] Michael Shirer. **IDC Spending Guide Forecasts Double-Digit Growth for Investments in Edge Computing**, 2022. 1

[17] Yi Liu, Chao Yang, Li Jiang, Shengli Xie, and Yan Zhang. **Intelligent edge computing for IoT-based energy management in smart cities**. *IEEE network*, **33**(2):111–117, 2019. 2

[18] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. **Vehicular edge computing and networking: A survey**. *Mobile networks and applications*, **26**(3):1145–1168, 2021. 2

[19] Ermanno Guardo, Alessandro Di Stefano, Aurelio La Corte, Marco Sapienza, and Marialisa Scatà. **A fog computing-based iot framework for precision agriculture**. *Journal of Internet Technology*, **19**(5):1401–1411, 2018. 2

[20] Simone Mangiante, Guenter Klas, Amit Navon, Zhuang GuanHua, Ju Ran, and Marco Dias Silva. **Vr is on the edge: How to deliver 360 videos in mobile networks**. In *Proceedings of the Workshop on Virtual Reality and Augmented Reality Network*, pages 30–35, 2017. 2

[21] European Telecommunications Standards Institute. **Multi-access Edge Computing (MEC); Framework and Reference Architecture**, 2020. 2, 6, 7, 9

[22] R Babu, K Jayashree, and R Abirami. **Fog Computing Qos Review and Open Challenges**. In *Research Anthology on Architectures, Frameworks, and Integration Strategies for Distributed and Cloud Computing*, pages 1147–1157. IGI Global, 2021. 2

[23] Inés Sittón-Candanedo, Ricardo S Alonso, Sara Rodríguez-González, José Alberto García Coria, and Fernando De La Prieta. **Edge computing architectures in industry 4.0: A general survey and comparison**. In *International Workshop on Soft Computing Models in Industrial and Environmental Applications*, pages 121–131. Springer, 2019. 5, 8

[24] ISO, IEC, and IEEE. **42010:2011 - Systems and software engineering - Architecture description**, 2017. 5

[25] John Dilley, Bruce Maggs, Jay Parikh, Harald Prokop, Ramesh Sitaraman, and Bill Weihl. **Globally distributed content delivery**. *IEEE Internet Computing*, **6**(5):50–58, 2002. 5

[26] Manjula Nanayakkara, Malka Halgamuge, and Ali Syed. **Security and privacy of internet of medical things (IoMT) based healthcare applications: A review**. In *International Conference on Advances in Business Management and Information Technology*, pages 1–18, 2019. 8

[27] Synergy Research Group. **Huge Cloud Market Still Growing at 34% Per Year**, 2022. 8, 43

[28] Alliance of Industrial Internet Edge Computing Consortium. **Edge Computing Reference Architecture 2.0**, 2017. 9

[29] FAR-EDGE. **FAR-EDGE Project H2020**, 2017. 10

[30] SAP Intel. **INTEL-SAP, IoT Joint Reference Architecture**, 2018. 10

[31] L. Canaran M. Tseng, T.E. Canaran. **Introduction to Edge Computing in IIoT**, 2018. 10

[32] Christine Ouyang Ashok Iyengar. **Edge Computing Reference Architecture**, 2020. 10

[33] Alexander Willner and Varun Gowtham. **Toward a Reference Architecture Model for Industrial Edge Computing**. *IEEE Communications Standards Magazine*, **4**(4):42–48, 2020. 10, 15

[34] Pasquale Pace, Gianluca Aloi, Raffaele Gravina, Giuseppe Caliciuri, Giancarlo Fortino, and Antonio Liotta. **An edge-based architecture to support efficient applications for healthcare industry 4.0**. *IEEE Transactions on Industrial Informatics*, **15**(1):481–489, 2018. 10, 13

[35] Mohammed Al-Khafajiy, Thar Baker, Carl Chalmers, Muhammad Asim, Hoshang Kolivand, Muhammad Fahim, and Atif Waraich. **Remote health monitoring of elderly through wearable sensors**. *Multimedia Tools and Applications*, **78**(17):24681–24706, 2019. 11

[36] Junxin Chen, Shuang Sun, Li-bo Zhang, Benqiang Yang, and Wei Wang. **Compressed sensing framework for heart sound acquisition in internet of medical things**. *IEEE Transactions on Industrial Informatics*, **18**(3):2000–2009, 2021. 13

[37] Getzi Jeba Leelipushpam Paulraj, Immanuel JohnRaja Jebadurai, Jebaveerasingh Jebadurai, and Nancy Emymal Samuel. **Cloud-Based Real-Time Wearable Health Monitoring Device Using IoT**. In *Computer Networks and Inventive Communication Technologies*, pages 1081–1087. Springer, 2021. 13

[38] Mario WL Moreira, Joel JPC Rodrigues, Vasco Furtado, Neeraj Kumar, and Valery V Korotaev. **Averaged one-dependence estimators on edge devices for smart pregnancy data analysis**. *Computers & Electrical Engineering*, **77**:435–444, 2019. 13

[39] Rahul Krishnan Pathinarupothi, P Durga, and Ekanath Srihari Rangan. **IoT-based smart edge for global health: Remote monitoring with severity detection and alerts transmission**. *IEEE Internet of Things Journal*, **6**(2):2449–2462, 2018. 13

[40] Md Abdur Rahman and M Shamim Hossain. **An internet-of-medical-things-enabled edge computing framework for tackling COVID-19**. *IEEE Internet of Things Journal*, **8**(21):15847–15854, 2021. 13

[41] Md Zia Uddin. **A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system**. *Journal of Parallel and Distributed Computing*, **123**:46–53, 2019. 13

[42] Fen Miao, Zeng-Ding Liu, Ji-Kui Liu, Bo Wen, Qing-Yun He, and Ye Li. **Multi-sensor fusion approach for cuff-less blood pressure measurement**. *IEEE journal of biomedical and health informatics*, **24**(1):79–91, 2019. 13

[43] G Enrico Santagati, Neil Dave, and Tommaso Melodia. **Design and performance evaluation of an implantable ultrasonic networking platform for the internet of medical things**. *IEEE/ACM Transactions on Networking*, **28**(1):29–42, 2020. 13

[44] Semir Salkic, Baris Can Ustundag, Tarik Uzunovic, and Edin Golubovic. **Edge computing framework for wearable sensor-based human activity recognition**. In *International Symposium on Innovative and Interdisciplinary Applications of Advanced Technologies*, pages 376–387. Springer, 2019. 13

[45] Amandeep Kaur and Ashish Jasuja. **Health monitoring based on IoT using Raspberry PI**. In *2017 International conference on computing, communication and automation (ICCCA)*, pages 1335–1340. IEEE, 2017. 13

[46] Neha Mathur, Greig Paul, James Irvine, Mohamed Abuhelala, Arjan Buis, and Ivan Glesk. **A practical design and implementation of a low cost platform for remote monitoring of lower limb health of amputees in the developing world**. *IEEE Access*, **4**:7440–7451, 2016. 13

[47] Chang Liu, Yu Cao, Yan Luo, Guanling Chen, Vinod Vokkarane, Ma Yunsheng, Songqing Chen, and Peng Hou. **A new deep learning-based**

**food recognition system for dietary assessment on an edge computing service infrastructure**. *IEEE Transactions on Services Computing*, **11**(2):249–261, 2017. 13

[48] Irina Valeryevna Pustokhina, Denis Alexandrovich Pustokhin, Deepak Gupta, Ashish Khanna, Kannan Shankar, and Gia Nhu Nguyen. **An effective training scheme for deep neural network in edge computing enabled Internet of medical things (IoMT) systems**. *IEEE Access*, **8**:107112–107123, 2020. 13

[49] Malathi Devarajan, V Subramaniyaswamy, V Vijayakumar, and Logesh Ravi. **Fog-assisted personalized healthcare-support system for remote patients with diabetes**. *Journal of Ambient Intelligence and Humanized Computing*, **10**(10):3747–3760, 2019. 13

[50] Admir Monteiro, Harishchandra Dubey, Leslie Mahler, Qing Yang, and Kunal Mankodiya. **Fit: A fog computing device for speech teletreatments**. In *2016 IEEE international conference on smart computing (SMARTCOMP)*, pages 1–3. IEEE, 2016. 13

[51] Minh Pham, Yehenew Mengistu, Ha Do, and Weihua Sheng. **Delivering home healthcare through a cloud-based smart home environment (CoSHE)**. *Future Generation Computer Systems*, **81**:129–140, 2018. 13

[52] Rajesh Gupta, Arpit Shukla, and Sudeep Tanwar. **Aayush: A smart contract-based telesurgery system for healthcare 4.0**. In *2020 IEEE International conference on communications workshops (ICC Workshops)*, pages 1–6. IEEE, 2020. 13

[53] Rojalina Priyadarshini, Rabindra Kumar Barik, and Harishchandra Dubey. **DeepFog: Fog computing-based deep neural architecture for prediction of stress types, diabetes and hypertension attacks**. *Computation*, **6**(4):62, 2018. 13

[54] Mohammad-Parsa Hosseini, Tuyen X Tran, Dario Pompili, Kost Elisevich, and Hamid Soltanian-Zadeh. **Multimodal data analysis of epileptic EEG and rs-fMRI via deep learning and edge computing**. *Artificial Intelligence in Medicine*, **104**:101813, 2020. 13

[55] Prateek Pandey and Ratnesh Litoriya. **Elderly care through unusual behavior detection: a disaster management approach using IoT and intelligence**. *IBM Journal of Research and Development*, **64**(1/2):15–1, 2019. 13

[56] Ali Hassan Sodhro, Zongwei Luo, Arun Kumar Sangaiah, and Sung Wook Baik. **Mobile edge computing based QoS optimization in medical healthcare applications**. *International Journal of Information Management*, **45**:308–318, 2019. 13

[57] Alaa Awad Abdellatif, Amr Mohamed, Carla Fabiana Chiasserini, Mounira Tlili, and Aiman Erbad. **Edge computing for smart health: Context-aware approaches, opportunities, and challenges**. *IEEE Network*, **33**(3):196–203, 2019. 13

[58] J Pena Queralta, Tuan Nguyen Gia, Hannu Tenhunen, and Tomi Westerlund. **Edge-AI in LoRa-based health monitoring: Fall detection system with fog computing and LSTM recurrent neural networks**. In *2019 42nd international conference on telecommunications and signal processing (TSP)*, pages 601–604. IEEE, 2019. 13

[59] Lynne A. Dunbrack. **Edge Computing: Transforming Healthcare by Increasing Resilience**, 2021. 15

[60] Jacques Durand Graham Bleakley Amine Chigani Robert Martin Brett Murphy Mark Crawford Shi-Wan Lin, Bradford Miller. **The Indsutrial Internet of Things Volume G1: Reference Architecture**. (1.9), 2019. 15

[61] Omer Ali, Mohamad Khairi Ishak, Muhammad Kamran Liaquat Bhatti, Imran Khan, and Ki-Il Kim. **A Comprehensive Review of Internet of Things: Technology Stack, Middlewares, and Fog/Edge Computing Interface**. *Sensors*, **22**(3):995, 2022. 19

[62] Ioan Ungurean. **Timing comparison of the real-time operating systems for small microcontrollers**. *Symmetry*, **12**(4):592, 2020. 21

[63] Eclipse Foundation. **IoT Edge Developer Survey Report**, 2021. 22, 25, 28, 29

[64] Istabraq M Al-Joboury and Emad H Al-Hemiary. **Performance analysis of internet of things protocols based fog/cloud over high traffic**. *Journal of Fundamental and Applied Sciences*, **10**(6S):176–181, 2018. 25

[65] Nitin Naik. **Choice of effective messaging protocols for IoT systems: MQTT, CoAP, AMQP and HTTP**. In *2017 IEEE international systems engineering symposium (ISSE)*, pages 1–7. IEEE, 2017. 25

[66] Upendra Tandale, Bashirahamad Momin, and Deva P Seetharam. **An empirical study of application layer protocols for IoT**. In *2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS)*, pages 2447–2451. IEEE, 2017. 25

[67] Jetendra Joshi, Vishal Rajapriya, SR Rahul, Pranith Kumar, Siddhanth Polepally, Rohit Samineni, and DG Kamal Tej. **Performance enhancement and IoT based monitoring for smart home**. In *2017 International Conference on Information Networking (ICOIN)*, pages 468–473. IEEE, 2017. 25

[68] Abdullah Ahmed Bahashwan, Mohammed Anbar, Nibras Abdullah, Tawfik Al-Hadhrami, and Sabri M Hanshi. **Review on common IoT communication technologies for both long-range network (LPWAN) and short-range network**. In *Advances on smart and soft computing*, pages 341–353. Springer, 2021. 27, 28

[69] Kento Takabayashi, Hirokazu Tanaka, and Katsumi Sakakibara. **Toward Dependable Internet of Medical Things: IEEE 802.15. 6 Ultra-Wideband Physical Layer Utilizing Superorthogonal Convolutional Code**. *Sensors*, **22**(6):2172, 2022. 27

[70] Nisrine Bahri, Safa Saadaoui, Mohamed Tabaa, Mohamed Sadik, and Hicham Medromi. **Wireless technologies and applications for industrial internet of things: a review**. *Advances on Smart and Soft Computing*, pages 505–516, 2021. 28

[71] Morghan Hartmann, Umair Sajid Hashmi, and Ali Imran. **Edge computing in smart health care systems: Review, challenges, and research directions**. *Transactions on Emerging Telecommunications Technologies*, **33**(3):e3710, 2022. 28, 30

[72] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. **An overview on edge computing research**. *IEEE access*, **8**:85714–85728, 2020. 30

[73] Djamel Eddine Kouicem, Abdelmadjid Bouabdallah, and Hicham Lakhlef. **Internet of things security: A top-down survey**. *Computer Networks*, **141**:199–221, 2018. 30

[74] Mohammad Aazam, Sherali Zeadally, and Eduardo Feo Flushing. **Task offloading in edge computing for machine learning-based smart healthcare**. *Computer Networks*, **191**:108019, 2021. 31

[75] Haifeng Lu, Chunhua Gu, Fei Luo, Weichao Ding, and Xinping Liu. **Optimization of lightweight task offloading strategy for mobile edge computing based on deep reinforcement learning**. *Future Generation Computer Systems*, **102**:847–861, 2020. 31

[76] Vishal A Thakor, Mohammad Abdur Razzaque, and Muhammad RA Khandaker. **Lightweight cryptography algorithms for resource-constrained IoT devices: A review, comparison and research opportunities**. *IEEE Access*, **9**:28177–28193, 2021. 31

[77] M. S. Turan K. McKay, L. Bassham and N. Mouha. **Report on Lightweight Cryptography (Nistir8114**. 2017. 31

[78] Abebe Abeshu Diro, Naveen Chilamkurti, and Neeraj Kumar. **Lightweight cybersecurity schemes using elliptic curve cryptography in publish-subscribe fog computing**. *Mobile Networks and Applications*, **22**(5):848–858, 2017. 32

[79] Yerrolla Chanti, Dr K Seena Naik, Rajesh Mothe, Nagendar Yamsani, and Swathi Balija. **A modified elliptic curve cryptography technique for securing wireless sensor networks**. *International Journal of Engineering & Technology*, **7**(8, December), 2018. 32

[80] Yumnam Winnie, E Umamaheswari, and DM Ajay. **Enhancing data security in IoT healthcare services using fog computing**. In *2018 International Conference on Recent Trends in Advance Computing (ICRTAC)*, pages 200–205. IEEE, 2018. 32

[81] Ricardo Martinez Santa and Fernando Martinez Santa Holman Montiel Ariza. **Secure Information Transmission Device Implemented on an Embedded System using 3DES and AES Algorithms**. *International Journal of Engineering Research and Technology*, **12**:1950–1956, 2019. 32

[82] Manuel Suárez-Albela, Paula Fraga-Lamas, and Tiago M Fernández-Caramés. **A practical evaluation on RSA and ECC-based cipher suites for IoT high-security energy-efficient fog and mist computing devices**. *Sensors*, **18**(11):3868, 2018. 32

[83] Nadar Anusha Xavier, Shanmuganathan Vimal, and V Jackins. **DSA-based secured communication in 5G vehicular networks using edge computing**. In *Computational Intelligence in Pattern Recognition*, pages 389–400. Springer, 2022. 32

[84] N Bhalaji. **Efficient and secure data utilization in mobile edge computing by data replication**. *Journal of ISMAC*, **2**(01):1–12, 2020. 32

[85] Sreeja Rajesh, Varghese Paul, Varun G Menon, and Mohammad R Khosravi. **A secure and efficient lightweight symmetric encryption scheme for transfer of text files between embedded IoT devices**. *Symmetry*, **11**(2):293, 2019. 32

[86] Yongjun Ren, Fujian Zhu, Jian Qi, Jin Wang, and Arun Kumar Sangaiah. **Identity management and access control based on blockchain under edge computing for the industrial internet of things**. *Applied Sciences*, **9**(10):2058, 2019. 32

[87] Shaoyong Guo, Xing Hu, Song Guo, Xuesong Qiu, and Feng Qi. **Blockchain meets edge computing: A distributed and trusted authentication system**. *IEEE Transactions on Industrial Informatics*, **16**(3):1972–1983, 2019. 32

[88] Yong Zhu, Chao Huang, Zhihui Hu, Abdullah Al-Dhelaan, and Mohammed Al-Dhelaan. **Blockchain-Enabled Access Management System for Edge Computing**. *Electronics*, **10**(9):1000, 2021. 32

[89] Tuan Nguyen Gia, L Qingqing, J Pena Queralta, Hannu Tenhunen, Zhuo Zou, and Tomi Westerlund. **Lossless compression techniques in edge computing for mission-critical applications in the IoT**. In *2019 Twelfth International Conference on Mobile Computing and Ubiquitous Network (ICMU)*, pages 1–2. IEEE, 2019. 33

[90] Kriti Bhargava, Stepan Ivanov, Chamil Kulatunga, and William Donnelly. **Fog-enabled WSN system for animal behavior analysis in precision dairy**. In *2017 International Conference on Computing, Networking and Communications (ICNC)*, pages 504–510. IEEE, 2017. 33

[91] Saurabh Singh, Shailendra Rathore, Osama Alfarraj, Amr Tolba, and Byungun Yoon. **A framework for privacy-preservation of IoT healthcare data using Federated Learning and blockchain technology**. *Future Generation Computer Systems*, **129**:380–388, 2022. 33

[92] Qiong Wu, Xu Chen, Zhi Zhou, and Junshan Zhang. **Fedhome: Cloud-edge based personalized federated learning for in-home health monitoring**. *IEEE Transactions on Mobile Computing*, 2020. 33

[93] Adnan Qayyum, Kashif Ahmad, Muhammad Ahtazaz Ahsan, Ala Al-Fuqaha, and Junaid Qadir. **Collaborative federated learning for healthcare: Multi-modal covid-19 diagnosis at the edge**. *arXiv preprint arXiv:2101.07511*, 2021. 33

[94] Saqib Hakak, Suprio Ray, Wazir Zada Khan, and Erik Scheme. **A framework for edge-assisted healthcare data analytics using federated learning**. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3423–3427. IEEE, 2020. 33

[95] Lanfang Sun, Xin Jiang, Huixia Ren, and Yi Guo. **Edge-cloud computing and artificial intelligence in internet of medical things: architecture, technology and application**. *IEEE Access*, **8**:101079–101092, 2020. 33

[96] Preeti Agarwal and Mansaf Alam. **Investigating IoT middleware platforms for smart application development**. In *Smart Cities—Opportunities and Challenges*, pages 231–244. Springer, 2020. 33

[97] Inc Amazon Web Services. **Connected Medical Devices with AWS IoT**, 2020. 33

[98] European Union Agency for Network and Information Security. *Baseline Security Recommendations for IoT.* 2017. 34

[99] Zhujun Xiao, Zhengxu Xia, Haitao Zheng, Ben Y Zhao, and Junchen Jiang. **Towards Performance Clarity of Edge Video Analytics**. 2021. 38

[100] Harshit Daga, Hobin Yoon, Ketan Bhardwaj, Harshit Gupta, and Ada Gavrilovska. **From back-of-the-envelope to informed estimation of edge computing benefits in minutes using castnet**. In *2019 IEEE International Conference on Fog Computing (ICFC)*, pages 165–174. IEEE, 2019. 39, 40

[101] Devki Nandan Jha, Khaled Alwasel, Areeb Alshoshan, Xianghua Huang, Ranesh Kumar Naha, Sudheer Kumar Battula, Saurabh Garg, Deepak Puthal, Philip James, Albert Zomaya, et al. **IoTSim-Edge: a simulation framework for modeling the behavior of Internet of Things and edge computing environments**. *Software: Practice and Experience*, **50**(6):844–867, 2020. 39

[102] Jonathan Hasenburg, Martin Grambow, and David Bermbach. **MockFog 2.0: automated execution of fog application experiments in the cloud**. *IEEE Transactions on Cloud Computing*, 2021. 39, 40

[103] Samvit Jain, Xun Zhang, Yuhao Zhou, Ganesh Ananthanarayanan, Junchen Jiang, Yuanchao Shu, Paramvir Bahl, and Joseph Gonzalez. **Spatula: Efficient cross-camera video analytics on large camera networks**. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 110–124. IEEE, 2020. 40

[104] Vinod Nigade, Lin Wang, and Henri Bal. **Clownfish: Edge and Cloud Symbiosis for Video Stream Analytics**. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 55–69. IEEE, 2020. 40

[105] Ji Lin, Wei-Ming Chen, Yujun Lin, Chuang Gan, Song Han, et al. **Mcunet: Tiny deep learning on iot devices**. *Advances in Neural Information Processing Systems*, **33**:11711–11722, 2020. 40

[106] Lanyu Xu, Arun Iyengar, and Weisong Shi. **CHA: A caching framework for home-based voice assistant systems**. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 293–306. IEEE, 2020. 40

[107] Huang Zeyu, Xia Geming, Wang Zhaohang, and Yuan Sen. **Survey on edge computing security**. In *2020 International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*, pages 96–105. IEEE, 2020. 40

[108] Letian Zhang and Jie Xu. **Fooling Edge Computation Offloading via Stealthy Interference Attack**. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 415–419. IEEE, 2020. 40

[109] Ayman Younis, Brian Qiu, and Dario Pompili. **Latency-aware hybrid edge cloud framework for mobile augmented reality applications**. In *2020 17th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, pages 1–9. IEEE, 2020. 40

[110] Yiqiang Chen, Xin Qin, Jindong Wang, Chaohui Yu, and Wen Gao. **Fedhealth: A federated transfer learning framework for wearable healthcare**. *IEEE Intelligent Systems*, **35**(4):83–93, 2020. 40

[111] Xiaojie Wang, Zhaolong Ning, Song Guo, and Lei Wang. **Imitation learning enabled task scheduling for online vehicular edge computing**. *IEEE Transactions on Mobile Computing*, 2020. 40

[112] Yueyue Dai, Du Xu, Ke Zhang, Sabita Maharjan, and Yan Zhang. **Deep reinforcement learning and permissioned blockchain for content caching in vehicular edge computing and networks**. *IEEE Transactions on Vehicular Technology*, **69**(4):4312–4324, 2020. 40

[113] Sabyasachi Gupta and Jacob Chakareski. **Lifetime maximization in mobile edge computing networks**. *IEEE Transactions on Vehicular Technology*, **69**(3):3310–3321, 2020. 40

[114] Zhuqi Li, Yuanchao Shu, Ganesh Ananthanarayanan, Longfei Shangguan, Kyle Jamieson, and Paramvir Bahl. **Spider: A Multi-Hop Millimeter-Wave Network for Live Video Analytics**. 2021. 40

[115] Vinod Nigade, Ramon Winder, Henri Bal, and Lin Wang. **Better Never Than Late: Timely Edge Video Analytics Over the Air**. In *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*, pages 426–432, 2021. 40

[116] Pengzhan Hao and Yifan Zhang. **EDDL: A Distributed Deep Learning System for Resource-limited Edge Computing Environment**. 2021. 40

[117] Massimo Gallo, Alessandro Finamore, Gwendal Simon, and Dario Rossi. **FENXI: Deep-learning Traffic Analytics at the edge**. 2021. 40

[118] Ji Lin, Wei-Ming Chen, Han Cai, Chuang Gan, and Song Han. **Mcunetv2: Memory-efficient patch-based inference for tiny deep learning**. 2021. 40

[119] Shruti Lall and Raghupathy Sivakumar. **Are Netflix Videos Edge-Cacheable? Exploration of a Deep Learning based Prefetching Strategy using a Real-World Dataset**. 2021. 40

[120] Pinyarash Pinyoanuntapong, Prabhu Janakaraj, Ravikumar Balakrishnan, Minwoo Lee, Chen Chen, and Pu Wang. **Edgeml: towards network-accelerated federated learning over wireless edge**. 2021. 40

[121] Eugene Kuznetsov, Yitao Chen, and Ming Zhao. **SecureFL: Privacy Preserving Federated Learning with SGX and TrustZone**. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*. IEEE, 2021. 40

[122] Hikmat Yar, Ali Shariq Imran, Zulfiqar Ahmad Khan, Muhammad Sajjad, and Zenun Kastrati. **Towards smart home automation using IoT-enabled edge-computing paradigm**. *Sensors*, **21**(14):4932, 2021. 40

[123] Abbas Mehrabi, Matti Siekkinen, Teemu Kämäräinen, and Antti yl Jski. **Multi-tier cloudvr: Leveraging edge computing in remote rendered virtual reality**. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, **17**(2):1–24, 2021. 40

[124] Tran Anh Khoa, Do-Van Nguyen, Minh-Son Dao, and Koji Zettsu. **Fed xData: A Federated Learning Framework for Enabling Contextual Health Monitoring in a Cloud-Edge Network**. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 4979–4988. IEEE, 2021. 40

[125] Xiangjie Kong, Kailai Wang, Shupeng Wang, Xiaojie Wang, Xin Jiang, Yi Guo, Guojiang Shen, Xin Chen, and Qichao Ni. **Real-Time Mask Identification for COVID-19: An Edge-Computing-Based Deep Learning Framework**. *IEEE Internet of Things Journal*, **8**(21):15929–15938, 2021. 40

[126] Li Pan, Lin Wang, Shutong Chen, and Fangming Liu. **Retention-Aware Container Caching for Serverless Edge Computing**. Technical report, 2022. 40

[127] Shreshth Tuli, Giuliano Casale, and Nicholas R Jennings. **PreGAN: Preemptive Migration Prediction Network for Proactive Fault-Tolerant Edge Computing**. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*, pages 670–679. IEEE, 2022. 40