TU WIEN Informatics

McEdgeChain: A lightweight blockchain ledger built upon a mission critical edge system

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Masterstudium Software Engineering Internet Computing

eingereicht von

Szabolcs Csörgő
Matrikelnummer 01527913

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Univ.Prof. Schahram Dustdar, Mag.rer.soc.oec. Dr.rer.soc.oec.
Mitwirkung: Univ.Ass. Ilir Murturi, BSc MSc

Wien, 8. August 2022

Szabolcs Csörgő	Schahram Dustdar

TU WIEN Informatics

McEdgeChain: A lightweight blockchain ledger built upon a mission critical edge system

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Master Software Engineering/Internet Computing

by

Szabolcs Csörgő
Registration Number 01527913

to the Faculty of Informatics

at the TU Wien

Advisor: Univ.Prof. Schahram Dustdar, Mag.rer.soc.oec. Dr.rer.soc.oec.
Assistance: Univ.Ass. Ilir Murturi, BSc MSc

Vienna, 8th August, 2022

Szabolcs Csörgő	Schahram Dustdar

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Szabolcs Csörgő

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 8. August 2022

Szabolcs Csörgő

Danksagung

Ich danke Ilir Murturi für seine Unterstützung und kompetente Beratung. Ohne seine Geduld und Ermutigung wäre dieses Papier nie erreicht worden.

Ich möchte Prof. Dustdar und der DSG-Gruppe dafür danken, dass sie alle IoT-Geräte für realistische Tests zur Verfügung gestellt haben. Es erlaubte mir, die Bottlenecks meiner Arbeit zu erkennen und Verbesserungen einzuführen.

Besonderer Dank geht an meine Frau für ihre allgemeine Unterstützung, Geduld und Verständnis. "Hinter jedem erfolgreichen Mann steht eine starke Frau."

Ich möchte mich bei meinen Eltern dafür bedanken, dass sie das passende Arbeitsumgebung erstellt haben, um meine Abschlussarbeit abzuschließen.

Zu guter Letzt, aber nicht zuletzt, bin ich Ihn dankbar, der mir die Fähigkeiten und die erwähnten Menschen gegeben hat. Deo gratias!

Acknowledgements

I would like to thank Ilir Murturi for his support and expert advice. Without his patience and encouragement, this paper would have been never accomplished.

I would like to thank Prof. Dustdar and the DSG group for providing all the IoT devices for realistic tests. It allowed me to detect the bottlenecks of my work and introduce improvements.

Special thanks go to my wife for her general support, patience, and understanding. "Behind every successful man, there is a strong woman."

I would like to thank my parents for creating the appropriate working environment to finish my thesis.

At last, but not least, I am grateful to the One who gave me the skills and the mentioned people. Deo gratias!

Kurzfassung

Edge Computing hat sich zu einer weit verbreiteten Technologie in verschiedenen realen Anwendungsszenarien entwickelt, einschließlich in Mission-critical Systemen (z. B. Rettungseinsätzen, Gesundheitssystemen usw.). Solche Systeme haben Anforderungen an niedrige Latenz und hohe Verfügbarkeit, während Zuverlässigkeit und Fehlertoleranz untrennbare Systemanforderungen sind. Traditionell werden geschäftskritische Systeme, wie sie im Gesundheitswesen eingesetzt werden, in der Cloud bereitgestellt und betrieben. Mit der wachsenden Datenmenge, die von Geräten des Internets der Dinge (IoT) (z. B. Sensoren, Aktoren usw.) erzeugt wird, wird die Cloud jedoch nicht in der Lage sein, strenge Anforderungen (d. h. Latenz, Verfügbarkeit und Zuverlässigkeit) zukünftiger kritischer Anwendungen (z. B. Verarbeitung von Sensordaten zur Gesundheit von Patienten in Echtzeit) zu erfüllen. So kann die Konvergenz zwischen Edge Computing und Blockchain mehrere betriebliche Herausforderungen lösen und beim Aufbau Mission-critical Systeme mit hoher Verfügbarkeit und einer zuverlässigen Umgebung für die Ausführung kritischer Operationen in der Nähe der Datenquelle bzw. am Rand des Netzwerks helfen.

Diese Arbeit zielt darauf ab, eine Edge-basierte Plattform aufzubauen, die die zuverlässige Verarbeitung der von IoT-Geräten generierten Gesundheitsdaten von Patienten gewährleistet. Die generierten Elektrokardiogramm-Daten (EKG) werden mit der Convolutional Neural Network (CNN) -Technik verarbeitet und in Arrhythmiekategorien eingeteilt. Die Arbeit stellt die Hypothese auf, dass der Aufbau eines Blockchain-Ledgers die Fehlertoleranz erhöht, ohne die Gesamtsystemleistung zu beeinträchtigen. Wir haben unsere Lösung anhand genauer Patienten-EKG-Daten evaluiert und auf einem realistischen Testbed aus mehreren Edge-Geräten getestet. In allen Anwendungsfällen messen wir Performance-Aspekte, Fehlertoleranz und analysieren Skalierbarkeitsaspekte. Die Ergebnisse sind vielversprechend, und die Fehlertoleranz des Systems wird durch Blockchain erhöht, ohne die Leistungsqualität zu beeinflussen.

Abstract

Edge computing has become a widely applied technology in various real-life use case scenarios, including those in mission-critical systems (e.g., rescue operations, healthcare systems, etc.). Such systems have low-latency and high availability requirements, while reliability and fault tolerance are inseparable system demands. Traditionally, mission-critical systems such as those within healthcare are deployed and operated on the cloud. However, with the growing amount of data produced by the Internet of Things (IoT) devices (i.e., sensors, actuators, etc.), the cloud won't be able to fulfill stringent requirements (i.e., latency, availability, and reliability) of future critical applications (e.g., processing in real-time patients health sensory data). Thus, the convergence between edge computing and blockchain can solve several operational challenges and assist in building mission-critical systems with high availability and a reliable environment for the execution of critical operations in proximity to the data source, respectively, at the edge of the network.

This thesis aims to build an edge-based platform that ensures the reliable processing of patients' health data generated by IoT devices. The generated Electrocardiogram (ECG) data is processed and classified into arrhythmia categories using the Convolutional Neural Network (CNN) technique. The thesis hypothesizes that building a blockchain ledger increases the fault-tolerance without affecting the overall system performance. We evaluated our solution using accurate patient ECG data and tested it on a realistic testbed comprising several edge devices. Throughout the use cases, we measure performance aspects, fault tolerance and analyze scalability aspects. The results are promising, and the system's fault tolerance is increased by blockchain without decreasing performance quality.

Contents

CHAPTER 1

Introduction

This chapter discusses the motivational scenario that forms the basis of the thesis. The problem statement is formulated to understand better what needs to be solved within this thesis. The research questions focus on different aspects of the system that need to be investigated. The solution approach describes the platform in a nutshell. The thesis structure provides the structure of the thesis.

1.1 Motivation

Edge computing is a new paradigm where substantial storage and computational resources, referred to as cloudlets, micro data centers, fog nodes, are located to the proximity of the data where it is generated. This paradigm aims to overcome the issues coming along with cloud computing [Sat17]. With the emerging number of IoT devices, it plays an essential role between the IoT devices and the cloud. The IoT devices are limited in computational power and storage; thus, sending all the data to the cloud results in high latencies and bandwidth usage. Hence, Edge computing can be thought of as a middleware between the IoT devices and the cloud to increase the performance of a system and mitigate the issues coming along with the cloud.

A mission-critical system is a system in which a failure or interruption comes with an unacceptable business or human cost [STÅK18]. In such systems, latency is among the most critical requirements. Thus, placing edge devices in such environments, respectively mission-critical systems, keeps the latency low and acts on processed data in a limited time constraint. A blockchain is a distributed immutable ledger that can store transactions. It started as a promising technology in the financial industry but nowadays goes beyond financial transactions. The main characteristic of blockchain is that the nodes operating in blockchain can perform without a trusted third party [EIPH18]. The main benefits provided by the characteristics of the blockchain are security, privacy, fault tolerance, and autonomous behavior.

Throughout this thesis, we consider emergency rooms respectively Intensive Unit Cares (ICUs) as mission-critical systems. Essentially, the emergency department (room) specializes in taking acute care of patients. Patients lying in the emergency room have no prior appointment; however, they need to be taken care of immediately in most cases. During the stay, various sensors and monitoring devices are attached to the patient's body providing the medical staff with the patient's health status. The collected data needs to be processed within a limited time constraint and, countermeasures need to be developed as the circumstances require. Otherwise, immutable severe consequences could occur that, in the worst cases, could lead even to live lost. Therefore, the velocity of processing the collected data goes hand in hand with the accurate knowledge of the patient's current health status. The more up-to-date data is present for doctors and nurses, the sooner can be acted in case of emergency, and the more lives can be saved.

Furthermore, it is also important to save the collected data for future analysis. Having an overall picture of a patient's health status as medical records can help doctors predict and better understand different diseases.

Figure 1.1: Emergency room with one bed [Eur17].

1.2 Problem statement

The mentioned scenario belongs to a category what is called a mission-critical system. Mission-critical (MC) systems have a common characteristic: reaction on the collected and processed data must happen within a limited time constraint to prevent loss. On the one hand, this means that the consumed time between collecting data and acting if certain conditions are met must be minimized. On the other hand, such systems require a high level of Quality of Service (QoS), like reliability, fault tolerance, to avoid tragedies. Also, an important aspect of the mentioned scenario is privacy since the data produced

within healthcare is extremely sensitive. The possibility to backup the data to a central cloud for further analysis needs to be addressed as well.

Summarizing up the described problems the following keywords and requirements are introduced:

- **Latency** - The data that the patient generates must be collected, processed, and acted on within a limited time constraint. The time constraint is not defined in an exact value; an ASAP policy will be followed within the thesis. The doctors can decide afterward whether the given results and performance are suitable for them.
- **Quality of Service** - Reliability and fault tolerance must be ensured. The system must stay reliable during the tests. All the data must be collected and evaluated to get a clear picture of the patients' health status. If some computational units become faulty during the evaluation, that must be also handled without a significant impact on the performance.
- **Privacy and security** - Sensitive data produced during the process must be handled properly. The health area operates on sensitive data that is coming from the patients. It needs to be ensured that no data can be stolen or checked by an unauthorized person. Secure communication between the different layers must be ensured as well.
- **Offloading** - Data must be backed up to a central cloud for further analysis. This backup helps to understand the collected data better and allows more complex evaluations. The offloaded data must arrive safely, secure from the edge devices on the cloud.

1.3 Research Questions

The motivation for this work is based on the above challenges and formulated as the following research questions:

(**RQ**1) *What is the maximum amount of patients that the system can handle simultaneously?*

The system needs to be tested in a simulated ICU room, where various sensors attached to patients produce the sensory data. An important measurement is to determine the number of patients the system can handle simultaneously. We focus on measuring two main system aspects such as i) latency and ii) reliability. The latency shows the time required to process sensory data and produce the final results. The reliability shows the data loss percent rate and success rate of processing sensory data.

(**RQ**2) *What is the relationship between the number of edge servers, fault tolerance, and the overall system performance?*

The number of edge servers applied in the ICU is increasing the fault tolerance of the system. The more edge servers are in charge; the more faulty nodes can be present in the platform. However, this implies also that more edge servers need to be involved in the consensus algorithm, which could lead to more overhead in terms of time. This question focuses on the number of edge servers and their impact on fault tolerance and latency.

(**RQ**3) *What is the latency overhead introduced by the blockchain technology compared to the non-blockchain solutions?*

Blockchain technology has numerous advantages, but it also has some disadvantages. Among the biggest disadvantage is the overhead introduced by the consensus algorithm, which impacts on the latency. To answer this question, the latency needs to be analyzed in the platform, where only one edge server is involved, and therefore no consensus algorithm needs to be performed.

(**RQ**4) *What is the impact of the faulty nodes on the performance?*

To have a reliable system the faulty nodes need to be handled in the system. In addition to that, we must analyze the overall impact of faulty nodes on the system performance. To achieve this goal a comparison is needed between two use cases: the ones, where no faulty nodes are present, and the ones, where faulty nodes are present.

1.4 Solution approach

The thesis aims to design and develop an appropriate platform for the scenario described above. The four defined keywords need to be addressed and the mentioned requirements should be met. Naturally, various approaches exist to reach the desired solution.

Traditionally, processing and streaming sensory and non-sensory data directly to the cloud is a viable solution. Infinite resources are available, the data can be easily backed up immediately, while no other middleware is needed. However, cloud computing comes with a huge latency, since the data must be forwarded to a location far away. This is the point where edge computing comes into play. Edge computing is a computational paradigm, that brings the computational resources closer to the place where the data is generated to bridge the gaps that come along with cloud computing e.g., high latency or bandwidth occupation. One key concept considering edge computing is offloading, which is well-researched in this area. The concepts and techniques can be reused to offload data from edge devices to the cloud for backup and further and more complex analysis.

Hence, Edge computing within the mission-critical application could ensure the low latency along with quick reactions to emergency cases, and the offloading is considered as well. However, the desired QoS needs to be still taken into account and has to be handled properly. One possible answer to ensure the QoS measurements is blockchain. The technology addresses fault tolerance, data immutability, privacy and security, and provides a solution to the mentioned bottlenecks.

The suggested solution is, therefore, to build a lightweight blockchain platform among the edge devices. The ledger will be constructed using the collected data from sensors. The edge devices processing the data coming from the sensors ensure low latency and provide offloading techniques to the cloud. The lightweight blockchain is taking care of the QoS and privacy. This paper hypothesizes that despite the introduced overhead of building a blockchain ledger, the latency could still be kept below a decent threshold, the system's reliability is maximized, privacy and security are ensured.

1.5 Thesis structure

The thesis has the following structure. Chapter 1 gives a brief introduction to the thesis, discusses the problem statement and the solution approach. Chapter 2 provides the background information, the theory that needs to be known to understand the thesis. Chapter 3 discusses the state-of-the-art solutions and describes how our approach is different from the solutions proposed by the research papers. Chapter 4 describes the technical solution, the architecture of the platform, and the solutions. In Chapter 5, the evaluation methodology is discussed, the use cases are introduced, and the testbed is described. Chapter 6 presents the results from the different use cases defined in the previous chapter and briefly discusses them. Chapter 7 provides the overall discussion of the platform and states the limitations. With Chapter 8, the thesis is concluded, the research questions are answered, and the future work is discussed.

CHAPTER 2

Background

This chapter provides the basic information that is necessary to understand the theory behind the thesis. First, the ICU, in general, is discussed, the dysfunction that needs to be detected is described, and the used dataset is introduced. The workload gives an understanding in a nutshell of the deep learning technique used in the platform. The main concepts of the platforms are discussed, such as Edge computing, blockchain, and the relation between those two. The consensus that forms the heart of blockchain technology is discussed in more detail because that forms the essential kernel of our platform and has a major impact on fault tolerance and performance.

2.1 Motivational scenario

The emergency department or ICU (Intensive Care Unit) is the place where critically ill patients without any prior appointment are handled. Due to the serious life risk present in such rooms, it is crucial to respond to any event occurring here immediately. To achieve this goal, all the patients staying in this department are constantly monitored by several sensors and monitoring devices.

In our case, each patient, that is laying in the room is represented by one monitoring device. In the focus of this thesis stands one specific health problem, i.e., arrhythmia. That means, the monitoring device attached to the patient body collects ECG signals, from which arrhythmia can be detected. The ECG signals are represented as individual heartbeats.

2.1.1 Arrhytmia

Arrhythmia is when an abnormal or improper heart beating rate is present by the patient, which leads to a failure in the blood pumping. If the heart rate is over a certain threshold, i.e., the heartbeats are faster than usual, it is called tachycardia. If the heart rate is under

a certain threshold, i.e., the heartbeats are slower than usual, it is called bradycardia. Arrhythmias are not related to the age of the person and, they can occur to anyone. Although, it is more common in people who suffer from high blood pressure, diabetes, and coronary artery disease. Some arrhythmias have no symptoms, while others can even cause death; therefore, the arrhythmia can be a serious problem and even life-threatening. The typical symptoms of arrhythmia are: premature beats skipped beats, dizziness, fatigue, fainting [AIH18].

Arrhythmias can be categorized by many factors, each identified by a pattern. However, the two major groups of arrhythmias are morpho-logical arrhythmia and rhythmic arrhythmias. Morpho-logical arrhythmia consists of arrhythmias formed by one irregular heartbeat. Rhythmic arrhythmia consists of arrhythmias of several irregular heartbeats [AIH18]. The thesis will check morpho-logical arrhythmias. Each arrhythmia will be analyzed and categorized into groups. To analyze the group of arrhythmias and derive further consequences from them is out of the scope of this thesis.

To represent the heart activity, an electrocardiogram is used. It is a graphical representation of the electrical waves generated by a heart. The information provided by ECG is heart rate, rhythm, morphology. Therefore, ECG is used by a cardiologist to detect abnormal heart rates produced by a patient. This detection has several difficulties. The main of them is the non-stationary nature of the heartbeat signal. It means ECG can identify the arrhythmia only if the arrhythmia occurs during the examination, monitoring. The arrhythmia can occur at any random time scale and any point of the day, and the symptoms of a disease do not show up all the time. As constant monitoring produces an enormous amount of data, it needs to be processed effectively [AIH18].

To test the platform's performance accurately, realistic scenarios using real data need to be constructed. The MIT-BIH database provides us with the mentioned data.

2.1.2 Dataset

The MIT-BIH Arrhythmia database is a set of data containing various records of an electrocardiogram. It was the first generally available dataset for evaluating arrhythmia detectors and for research purposes as well. In 1975 the researchers of MIT recognized the need for long-term ECG data for the research, so they began to collect, analyze and annotate ECG data obtained from the Beth Israel Hospital (BIH), hence the name MIT-BIH [MM01].

The MIT-BIH Arrhythmia Database contains 48 half-hour excerpts of two-channel ambulatory ECG recordings obtained from 47 subjects studied by the BIH Arrhythmia Laboratory between 1975 and 1979. Twenty-three recordings were chosen at random from a set of 4000 24-hour ambulatory ECG recordings. The data was collected from a mixed population of inpatients (about 60%) and outpatients (about 40%) at Boston's Beth Israel Hospital. The remaining 25 recordings were selected from the same set to include less common but clinically significant arrhythmias that would not be well-represented in a small random sample [MM01].

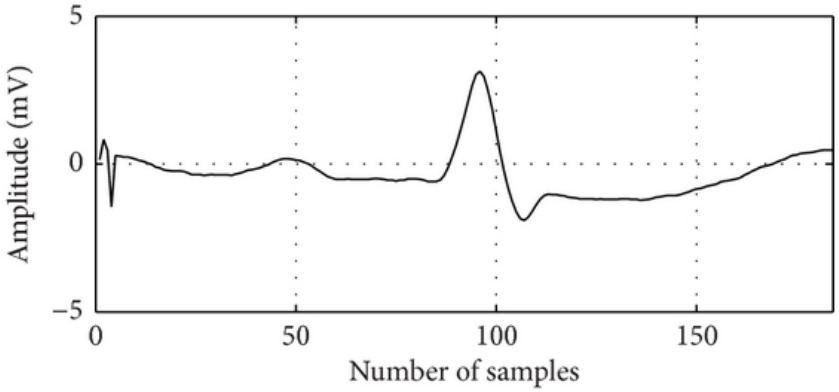

Figure 2.1: Heartbeat categorized as normal according to the AAMI standard [DA14].

The recordings were digitized at 360 samples per second per channel with 11-bit resolution over a 10mV range. Two or more cardiologists independently annotated each record, disagreements were resolved to obtain the computer-readable reference annotations for each beat (approximately 110,000 annotations in all) included with the database [MM01]. The data in the MIT-BIH Arrhythmia Database is in raw format that needs to be pre-processed and segmented to make it usable for our thesis. Fortunately, this does not need to be done by us because a proper data extraction already exists.

Kachuee et al. [KFS18] created an extraction of the database, where ECG lead II was used as the input, that was re-sampled to the sampling frequency of 125 Hz. Each sample refers to a heartbeat, and the total number is 109 446. For the classification of the heartbeats, five different categories were used following the Association for the Advancement of Medical Instrumentation (AAMI) EC57 standard. The mapping is shown in Table 2.1.

2.1.3 Workload

The workload of the system is to identify arrhythmias based on heartbeat signals. The platform should be able to classify each heartbeat signal into different arrhythmia categories described in Table 2.1. For that, a deep learning technique is used. In this case, a Convolutional Neural Network is built. In this subsection, CNN is discussed, to an extent, that is feasible to understand its main concepts. It is worth mentioning that although in the explanations of CNN, the input is usually referred to as an image, any arbitrary data can be substituted as input.

A Convolutional Neural Network (CNN) is a deep learning algorithm. To describe it in a nutshell, the algorithm assigns importance (learnable weights and biases) to the various aspects of the input (mostly image) to differentiate objects contained by the input. The pre-processing costs of this technique are lower than the costs of other

Category	Annotations
N	Normal Left/Right bundle branch block Atrial escape Nodal escape
S	Atrial premature Aberrant atrial premature Nodal premature Supra-ventricular premature
V	Premature ventricular contraction Ventricular escape
F	Fusion of ventricular and normal
Q	Paced Fusion of paced and normal Unclassifiable

Table 2.1: Summary of mappings between beat annotations and AAMI EC57 categories [KFS18].

classification algorithms, which makes this technique popular. The CNN applies filters on the input to capture the spatial and temporal dependencies. Simply put, CNN is trained to understand the sophistication of the input better. It can reduce the input into a form that is easier to comprehend without losing features that are critical to getting a good prediction. That makes the CNN scalable with acceptable pre-processing costs [Sah18].

The first layer of CNN is the convolutional layer. The convolutional layer is constructed by applying a kernel. A kernel is a matrix that is applied on the top of the input matrix. It is shifted with the given stride on the top of the input performing a matrix multiplication within each frame. The purpose of the kernel is to capture the high-level features and dependencies of the input. For example, if an image represents the input, those properties could be edges or color gradients. With added convolutional layers, more sophisticated features of the input can be detected. [Sah18]

After the convolutional layers, a pooling layer is next. It is used to reduce the spatial size of the input and decrease the computational power needed to process it. Two types of pooling are known: max pooling and average pooling. The former returns the max value from the portion of the input the latter returns the average. The convolutional layer, together with the pooling layer, forms the i-th layer of the network. The well-chosen number of such layers can increase the understanding of the input and hence can lead to better classification [Sah18].

The last piece of CNN is the fully connected layer. The fully connected layer is usually a cheap way of learning non-linear combinations of the features represented by the output processed by the series of convolutional and pooling layers. The output is first flattened and fed to a feed-forward neural network and, backpropagation is applied to each iteration

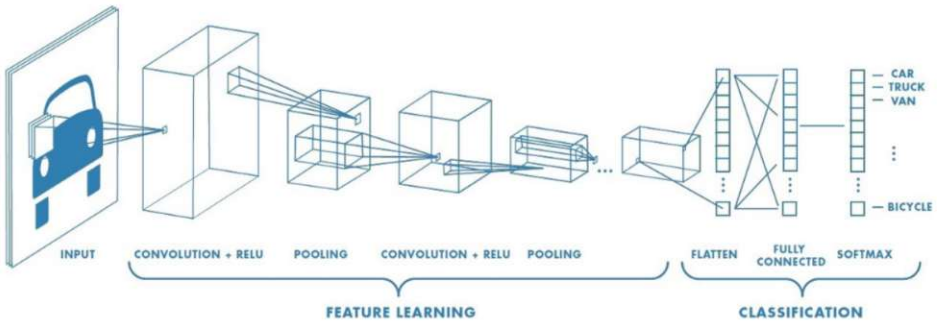

Figure 2.2: Convolutional Neural Network overview [Sah18].

of training. After a given amount of epochs, the CNN can distinguish certain features of the input and classify them into classification groups. The output is a classification vector that contains a probability for each classification category [Sah18]. Therefore, to solve the arrhythmia categorization problem convolutional network is built. The network can classify each heartbeat into different categories of arrhythmia with a certain likelihood. The category with the highest probability is chosen as the prediction of the system.

2.2 Main concepts of the platform

Within this thesis, a new platform is created. The platform is named McEdgeChain, which stands for a blockchain solution over an Edge computing environment applied in a mission-critical (Mc) area. The further sections describe the concepts in detail, focusing on knowledge that is required for the platform.

2.3 Edge computing

In Edge computing, the most important term is physical proximity, which has been overseen by the endless connectivity offered by the internet. The physical proximity is characterized by low latency, low jitter, and high bandwidth. However, the question "How close is physically enough" can't be answered in general because it depends on many factors [Sat17]. Although, in general, deploying edge servers in the proximity of the network could bring the following benefits as discussed in [Sat17]:

- *Highly responsive cloud services* - the proximity allows to achieve low end-to-end latency, high bandwidth, and low jitter to services located on the edge server.
- *Scalability via edge analytics* - the data could be analyzed and pre-processed on the edge servers and, the much smaller extracted information and metadata must be transmitted to the cloud.

- *Privacy-policy enforcement* - by serving as the first point of contact in the infrastructure for sensor data, an edge server can enforce the privacy policies before sending the data to the cloud.
- *Masking cloud outages* - if a cloud service becomes unavailable due to network failure, cloud failure, or a denial-of-service attack, a fallback service on a nearby cloudlet can temporarily mask the failure.

2.4 Blockchain

Blockchain as technology was proposed in 2008 and was implemented in 2009 through the famous cryptocurrency Bitcoin. The technology could be described as a ledger, in which all the transactions are stored in chains of blocks. The blocks are added by autonomous, decentralized nodes integrating technologies such as cryptographic hash, digital signature, and distributed consensus mechanism. Hence, blockchain applied in certain situations can improve efficiency. Although blockchain was meant to use for cryptocurrencies, several areas exist where blockchain technology can be applied since it allows the creation of verified data ledgers without third-party [ZXD+18].

The main building blocks of a Blockchain are [EIPH18]:

- *Transactions*, that are signed pieces of information created by the participating nodes in the network that are broadcasted to the rest of the network.
- *Blocks*, that are collections of transactions appended to the blockchain after being validated.
- *Blockchain* is a ledger (sequence of blocks) of all the created blocks, each block containing a reference to the previous block.
- *Consensus* mechanism is used to decide which blocks are added to the blockchain.

Hence, blockchain provides trustless, decentralized peer-to-peer data storage that is synchronized around all the participants. The data is organized as a sequence of blocks, where each block is connected to the previous block. The connection is immutable because of the cryptographic hash function used by each block. The very first block of the chain, called the genesis block, is a dummy block containing no relevant information. Let's see that no restriction on the data exists; thus, any arbitrary data can be stored in a blockchain [IPG+20]. The immutability is ensured by the fact that each hash of the block contains the previous block's hash. Hence, each new block committed to the chain contains all the information of the subchain indirectly before the block. To alter one block in the chain, the whole chain needs to be changed.

To decide which block should be added to the chain, the P2P network needs to reach a consensus. The new proposed block should be validated and approved by the other

nodes to extend the chain. The consensus allows the system to operate without a trusted third party.

The main characteristics of blockchain technology are: [IPG+20]

- *Decentralization* – the data is spread over multiple participants, and the integrity of the data is provided by many decentralized parties;
- *Immutability* - information integrity is created by immutable ledgers, blocks rely on each other.
- *Security* – the elimination of single-point failure and usage of public key infrastructure makes this architecture more secure.
- *Efficiency* - distributed transactions in the database make the verification more transparent so efficiency is reflected in terms of cost settlement speed, and risk management.
- *Transparency* - transactions details are shared between all users involved in a blockchain network.

In general, the blockchain platforms can have 3 different types as specified in [EIPH18]:

- *Public permissionless blockchains*, that serves a "low trust" environment enabling everyone to join the network and participate in the consensus protocol. Besides that, the whole ledger of transactions can be read by any participant. Examples of such a platform are Bitcoin or Ethereum.
- *Private blockchains*, where all the participants are added by a trusted central organization. Therefore, access to the network is managed by a central organization, and the consensus protocol is controlled by them as well. Examples of such a platform are Multichain or Eris.
- *Public permissioned blockchains*, that provide a hybrid model between the two aforementioned platforms. The main characteristic of such a blockchain is, that the access and the control of consensus protocol are managed by a single set of nodes. Examples of such platforms are Hyperledger Fabric or Ripple.

2.5 Blockchain & Edge computing

As was stated earlier, blockchains demonstrated a great potential in many areas, not just in the financial sector. It is worth to consider using blockchain in areas, where usually a centralized client/server architecture is used, such as in IoT ecosystems. The used blockchain architecture can mitigate the delays and failures caused by the centralized architecture [EIPH18]. El et al. [EIPH18] focus on three challenges, that can be solved by introducing blockchain technology to the Edge computing paradigm.

Confidentiality and integrity: Applying blockchain technology among devices makes it much harder to corrupt them by i) using immutable cryptographically verifiable data that is shared by all the participants in the network, and ii) validating the integrity of the network transactions before accepting them. Also, the linked nature of the blocks in a blockchain makes the schema hard to break. When it comes to confidentiality, some blockchains allow the encryption of the payload data to add another layer of security.

Autonomous behavior: Blockchains offer functionality allowing the management of infrastructure for autonomous agents in the form of smart contracts, which are self-executing programs residing on the blockchain itself. Smart contracts encapsulate business logic and conditions determining when a contract is going to be executed. The behavior of a device can be specified by a set of smart contracts that allow it to interact with the rest of the network.

Fault tolerance: The peer-to-peer nature of blockchain technology increases fault tolerance and availability of the system as the failure of some nodes will not paralyze the whole network. The decentralized architecture of blockchain also allows for lighter, faster, more reliable, and secure communication between nodes.

2.6 Consensus algorithm

The nodes maintaining the blockchain need some common agreement to decide which of the nodes is allowed to append the next block to the chain. For this purpose, consensus algorithms are used. In basic terms, a consensus algorithm is a technique to conclude inside of a group. The nodes participating in the decision need to reach a consensus. Not the individual choices are important, but the choice that works best for all the nodes [PKP19]. This consensus guarantees that no third party is needed for supervision, and hence the nodes can operate autonomously, and uniformity is provided. Two kinds of consensus algorithms rule the blockchain world: proof-based and voting-based. The former is used mainly in public blockchains and ensures the consensus by providing that a specific work (mathematical puzzle solved, more stake in blockchain, more space) has been done. The latter is preferred in private blockchains, and the consensus is made by collecting the votes from the nodes [PKP19].

In distributed systems, no perfect consensus protocol exists. If a protocol is chosen, a trade-off must be made between consistency, availability, and partition fault tolerance. Besides the mentioned properties, the Byzantine Generals Problem needs to be addressed as well. This problem assumes that malicious nodes will appear in the systems that can falsify the consensus algorithm. The most famous consensus protocols are provided below to give an overview [ZL19].

2.6.1 Proof of work (PoW)

PoW is a consensus algorithm adopted by Bitcoin or Ethereum. The main idea of this consensus algorithm is computational competition. The nodes participating in the

algorithm need to solve a challenging cryptographic puzzle, and the node that solves this puzzle first has the right to attach the new block to the chain. The puzzle can be solved by adjusting a nonce value to get the correct format of the hash. It is feasible to overthrow one block in the chain, but the more blocks exist in the chain, the more computational power is needed to replace fractions of the chain. PoW belongs to the probabilistic-finality consensus protocols because it ensures eventual consistency [ZL19].

2.6.2 Proof of Stake (PoS)

In PoS, selecting each round of nodes that creates a new block depends on the held stake rather than the computational power. In PoS, compared to the PoW, the key is not to adjust the nonce-value but to have a proper amount of the stakes. The PoS algorithm is energy-saving because it uses far less computational power than the PoW [ZL19].

2.6.3 Delegated Proof of Stake (DPoS)

The principle of the DPoS is to let the stakeholders elect dedicated nodes that will serve as block creators in the systems. The creation of a new block is delegated to nodes, thus reducing the energy consumption of the stakeholders. In DPoS, if a delegated node can not generate a new block, it will be dismissed and replaced by a new node selected by the stakeholders. To reach the consensus, the votes are taken into account fairly and democratically. This consensus algorithm, compared to the former is low-cost and high-efficiency. Examples adopting this technique are BitShares, EOS, etc. [ZL19]

2.6.4 Ripple

Ripple is an open-source payment protocol. The transactions are initiated by clients and spread across the network to validating nodes. Each validating node holds a list of trusted nodes UNL (Unique Node List). Nodes contained by the UNL list can vote on transactions they support. The proposed transaction will get a vote if the same transaction exists in the local transaction set. If a transaction gets 50% support, it is allowed to step into the next round. The threshold is increased in each round and, in the last round, a transaction with more than 80% of votes will be added to the blockchain ledger. Ripple is an absolute-finality consensus protocol [ZL19].

2.6.5 Practical Byzantine Fault Tolerance (PBFT)

The Practical Byzantine Fault Tolerance consensus algorithm has low algorithm complexity and high practical use in the distributed systems. In PBFT five phases exist: request, pre-prepare, prepare, commit, reply. The primary node that is responsible for generating the next block sends its proposal to the other nodes. The message needs to pass through the five phases, each time validated by nodes. If enough message is received in the current phase, the proposed block enters the next phase. It is then distributed again with the new phase header. If the block reaches the final state, we know that all the nodes agree

on the validity. In that case, the block can be safely added to the chain. PBFT ensures a commons state among the nodes by taking consistent action in each round. Because of the strong consistency, PBFT is an absolute-finality consensus protocol [ZL19].

Property	PoW	PoS	DPoS	PBFT	Ripple
Finality Type	Probab.	Probab.	Probab.	Abs.	Abs.
Fault tolerance	50 %	50 %	50 %	33 %	20 %
Power consumption	Large	Less	Less	Negligible	Negligible
Scalability	Good	Good	Good	Bad	Good
Application	Public	Public	Public	Permissioned	Permissioned

Table 2.2: Main consensus protocols comparison [ZL19].

2.6.6 Bully Algorithm

The bully algorithm is not a typical blockchain consensus algorithm. The bully algorithm is an election algorithm, where a set of peers need to elect a specific node, the coordinator. If the coordinator fails, the peers should be able to detect the failure and elect a new coordinator in the system [RN10]. The coordinator is responsible to perform tasks in the platform. It can be a scheduling task or to act as a worker and execute jobs.

Bully algorithm was proposed by Garcia-Molina in 1982. The algorithm presumes basic assumptions [RN10]:

- the system is synchronous, and the coordinator failure is tracked by timeout mechanism
- each node has a unique identifier
- every node is discovered by every other node
- the nodes does not know the current up and down status of the other nodes
- in the election the node with the highest id is selected as a coordinator

The bully algorithm uses 3 types of messages [RN10]:

- **election** - this type of message is used to initiate an election
- **ok** - this message is use as an acknowledgement to a certain message
- **coordinator** - the node, that is up and has the highest ID sends it to the other nodes, to announce the victory of the election

The procedure of the bully algorithm is the following: [RN10]

1. A node detects, that the coordinator does not respond in a certain timeout
2. The node sends an election message to all the nodes that have higher IDs. If none of the nodes with the higher ID acknowledges this message, that means that the current node is the active node with the highest ID, and hence it sends the coordinator message to the other nodes.
3. If the node gets an acknowledgment to the election message, that means an active node with a higher ID exists in the system. In that case, the node waits for the coordinator message from another node.

CHAPTER 3

Related work

In this chapter, different papers are examined and analyzed to get an overview about the state of the art solutions and the related work. The papers are grouped by topic. The first group checks the papers about Edge computing in healthcare, the second group the Edge computing in mission-critical systems, and the third the relation between blockchain in Edge computing and healthcare. At last, our approach is described and what differentiates it from the afore-mentioned approaches.

3.1 Edge computing in Healthcare

Several attempts have been made to apply the Edge computing paradigm in the healthcare area. Akrivopoulos et al. [ACTA17] sketch the usage of fog (edge) computing in healthcare. Use case scenarios are provided, the important requirements towards such system are described and, a prototype is developed focusing on ECG analysis. Gia et al. [GJR+15] realize a healthcare system based on fog computing providing a detailed description of the services and requirements towards such system. Chen et al. [CLH+18] propose a healthcare system realized on edge devices where the focus is placed on determining the risk level of the patients based on the collected data and reallocating the resources accordingly. In the work of Rahmani et al. [RGN+18] smart gateways are used as edge devices that take care of data-preprocessing, compression, encryption, and revealing the emergencies using early warning scores. Azimi et al. [AAR+17] analyze an existing healthcare system MAPE-K and suggest a novel computing architecture suitable for hierarchical partitioning and execution of machine learning-based data analytics and a closed-loop management technique capable of autonomous system adjustments concerning patient's condition.

3.2 Edge computing in mission-critical system

Mission-critical systems provide also opportunities to research the benefits of using Edge computing in that area. Skarin et al. [STÅK18] define a testbed for a mission-critical system Edge computing environment and justifies the applicability of Edge computing in the mission-critical systems with 5G. Zhang et al. [ZF15] elaborate on software-defined networks, classify IoT and focus on the communication and latency among them. Xu et al. [XYYZ19] propose a software-defined mission-critical wireless sensor networking, define the mission-critical task and establish an algorithm to make offloading decisions.

3.3 Blockchain in Edge computing & Healthcare

Blockchain as an emerging technology is involved in Edge computing as well. El et al. [EIPH18] describe the benefits of using blockchain among IoT devices and develop a decision framework on whether the technology is needed in the current situation. Esposito et al. [EDST+18] sketch the usability of blockchain in healthcare and the challenges as well. Stanciu et al. [Sta17] use a Hyperledger Fabric as a blockchain platform on a supervision level (cloud) and edge device to control the devices on the lower level. Rabah et al. [Rab17] research how the introduction of blockchain in healthcare could improve the security, data privacy, and interoperability of medical records. Ekblaw et al. [EAHL16] develop a prototype of blockchain for electronic health records and medical research data addressing the following problems: fragmented, slow access to medical data, system interoperability, patient agency. Zhang et al [ZSWL18] provide use cases where blockchain could be used in healthcare and based on them a D-App blockchain platform is designed and developed. The work also describes in detail the issues at design time and proposed solutions to them. Buchman [Buc16] describes how blockchain can be used to achieve Byzantine fault tolerance.

3.4 Our approach

What differentiates our approach from the above-mentioned works is that we apply blockchain technology for medical data not on the cloud level but on the edge level to ensure appropriate QoS. A blockchain ledger is built from sensor data rather than from whole medical records. We hypothesize that blockchain used on the lower levels can ensure the reliability, security, and fault tolerance of the system and at the same time keep the latency of such mission-critical applications under an appropriate threshold.

CHAPTER 4

McEdgeChain

In this chapter, the platform is described in general. The problem setup is discussed together with the actors of the system from which the requirements are derived. The major decisions regarding the blockchain and the communication are described. At last, the implementation details, which are the bottlenecks of the system, are discussed.

4.1 Problem setup

The first thing to do is to get an overview of the problem setup. The different actors are discussed and how they relate to each other. It is necessary to derive all the requirements for the platform. As it is visible from Figure 4.1 three main actors are present in the platform.

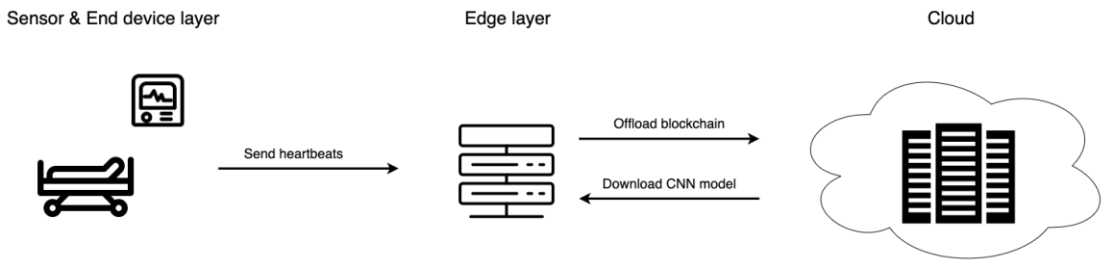

Figure 4.1: Problem setup.

4.1.1 Sensor & End device layer

This layer is responsible for monitoring the patients that are present in the ICUs. In this particular case, all the sensors are ECG sensors attached to the patient's body. The computational capacity of the sensors is marginal and, it is not able to do any analysis or

evaluation. The collected data is raw that needs a comprehensive interpretation, which is measurable. It is the point where the end devices come into play.

The end devices are connected to the sensors and take the responsibility to interpret the collected data for further evaluation. In this paper, the terms end device and monitoring device refer to the same expression. We also won't separate the sensor layer from the end device layer. That means the transmission time is omitted between those two layers and does not play any role regarding the performance.
This layer serves as a starting point for our platform. The preprocessed and segmented MIT-BIH testing dataset is broadcasted through end devices to the edge servers for evaluation.

4.1.2 Edge layer

In this paper, the edge servers are connected to the end devices and receive the data gathered by them. The edge server takes the responsibility to process the data and build the blockchain ledger. On the other hand, they are also connected to the cloud. The blockchain ledger needs to be offloaded to the cloud for backup purposes and further analysis. Also, to improve the pre-trained model that makes predictions, the edge server should be able to download the newly generated model by the cloud on regular basis.

4.1.3 Cloud

In this paper, the cloud will serve as a data center, where the ledger built by edge servers must be offloaded for further analysis. It also trains the CNN model that predicts arrhythmia. The blockchain coming from edge servers is used to improve the CNN model by newly performing training.

4.2 Requirements towards the platform

In this section the requirements towards the different parts of the platform are discussed and described.

4.2.1 Sensor & End device

ID	Description
ReqED0	Having 18 subset from testing dataset of heart beats in .csv format, each of them referring to half-hour period
ReqED1	Collect data from the corresponding .csv file
ReqED2	Create a data format for broadcasting
ReqED3	Broadcast the created data in each 1,496 sec to a dedicated channel

Table 4.1: Requirements towards end device.

ID	Description
ReqES0	Be able to build a P2P network from edge servers
ReqES1	Be able to add new edge servers to the network
ReqES2	Receiving data from the end devices
ReqES3	Decide which of the nodes should calculate the new block
ReqES4	Predict the arrhytmia based on the collected data
ReqES5	Transform the predicted data together with the samples into a block
ReqES6	Validate the proposed block
ReqES7	Append the new block to the chain
ReqES8	Offload the chain to the cloud
ReqES9	Receive the notification from the cloud, when a new model is ready
ReqES10	Download the new model from the cloud
ReqES11	Save the blockchains for performance analysis

Table 4.2: Requirements towards edge servers.

The end devices are broadcasting the data to the edge servers. Therefore, the first requirement is to have appropriate data. We are using the testing dataset extracted by Kachuee et al. [KFS18]. The data set needs initial pre-processing. It needs to be split into 18 subsets of heartbeats, each subset referring to a half-hour period. It is organized to *.csv* files to make it readable for the end devices. Each *.csv* contains 1209 rows representing 1209 heartbeats.
One row contains 188 columns, which refers to 187 samples, and one additional column, which holds the category of the heartbeat. Because the sampling rate is 125 Hz, each 8 ms a sample is made. So, once the data is loaded, a Data Transfer Object (DTO) is created, containing the following attributes: the end device ID, that in this case is represented by a patient, the heartbeat ID, and all the belonging sample with the sample value and the order in which it should be organized to get the heartbeat.

This DTO is broadcasted to the channels in each 1,496 sec because that time interval covers one heartbeat with 187 samples each sample taken in 8 ms. Each time a whole heartbeat is sampled, it is sent to the edge servers to process and evaluate it.

4.2.2 Edge server

The edge servers are the core part of the platform. They are responsible for the main work done by McEdgeChain. First of all, because the platform operates on multiple servers, a proper network should be built. We are using a P2P network, where each node should know about the others. It is also essential to control the addition of a new edge server. Since the blockchain is private, it needs to be ensured that no unknown node can participate in the block creation and the consensus. To fulfill this requirement, a message from a trusted host is broadcasted to existing nodes, with all the necessary information about the new edge server. After the data is broadcasted to the other nodes, the connections are established, and the new node is considered as a participant. The

ID	Description
ReqCL0	Init the CNN model at the first run
ReqCL1	Be able to receive the offloaded blockchains
ReqCL2	Be able to refine the existing CNN model based on the new data
ReqCL3	Notify edge server once the CNN model is ready
ReqCL3	Be able to provide the new model for the edge servers

Table 4.3: Requirements towards Cloud.

data coming from the end devices need to be handled properly. The heartbeats come in *json* format, so it needs a transformation into a DTO. Afterward, a new block should be created and proposed.

The platform should decide which of the given nodes should propose the next block to the chain. It should be consistent, so each node should have the same node as the next proposer. On the other hand, it needs to be simple enough to reduce the overhead as much as possible. The workload that needs to be solved is the arrhythmia prediction. To make precise predictions, a neural network model is built based on the training set, which is used to decide in which arrhythmia category should be the given heartbeat categorized. The predicted category, together with the samples, should be included in the next block. If the block is ready, it should be broadcasted to other nodes. The nodes should validate the incoming data, i.e. it contains the expected amount of heartbeats, the predicted arrhythmia category, whether the proposed block comes from the elected node and the data is in proper format. If it is valid, the block can be considered as the correct new block and can be added to the chain. For each patient a new chain is created, thus at the end the created amount of blockchains equals to the amount of patients.

After a certain amount of time, the blockchains created by the edge servers should be offloaded to the cloud for further analysis. That job is done periodically, and each time the whole blockchain is offloaded. The cloud should send a notification when the CNN model that predicts the arrhythmia is refined (trained with the offloaded data) and ready to be deployed on the edge servers. In that case, the edge server should download the model and reload it to improve prediction accuracy. The created blockchains should be saved for further analysis. Each patient should have at the end a *json* file, where the created blockchain should be visible. This *json* can be used in the future to analyze the correctness of the predictions and the performance of the platform.

4.2.3 Cloud

The cloud is the part of the platform that is in direct connection with edge servers. The cloud has a high computational capacity, so it is responsible for initializing the CNN model. That means the CNN model should be constructed and trained with the prepared training data. The cloud should be able to receive the blockchains periodically offloaded by the edge servers. This data is backed up in the cloud for further analysis and to refine

the model. If enough data from the edge server is available, the cloud should be able to refine the existing model with the new data. The refining and the initialization are the most computationally intensive tasks in this platform.

Once the refined model is ready, the cloud should notify all the edge servers. The cloud should be able to provide an opportunity for edge servers to download the refined model if it is requested.

4.3 Blockchain

Because the hospital where the platform is applied could be considered as a trusted client, and there is no need to participate in the network by anyone without permission from the trusted client, the type of blockchain used in this thesis is: **Private blockchain**.

Since the platform developed within this thesis is a private blockchain, a closer look is taken into two consensus techniques PBFT and Ripple. Regarding the fault tolerance, the PBFT performs better than the Ripple. The PBFT can tolerate at most f number of malicious or faulty nodes if the the whole system consists *3f+1* nodes. That means the normally behaving nodes should exceed an amount of *2f+1* [CL+99]. The fault tolerance of Ripple is only 20 %, i.e., Ripple can tolerate Byzantine Problem in 20 % of nodes in the entire network without affecting the correct result of consensus [SYB+14].

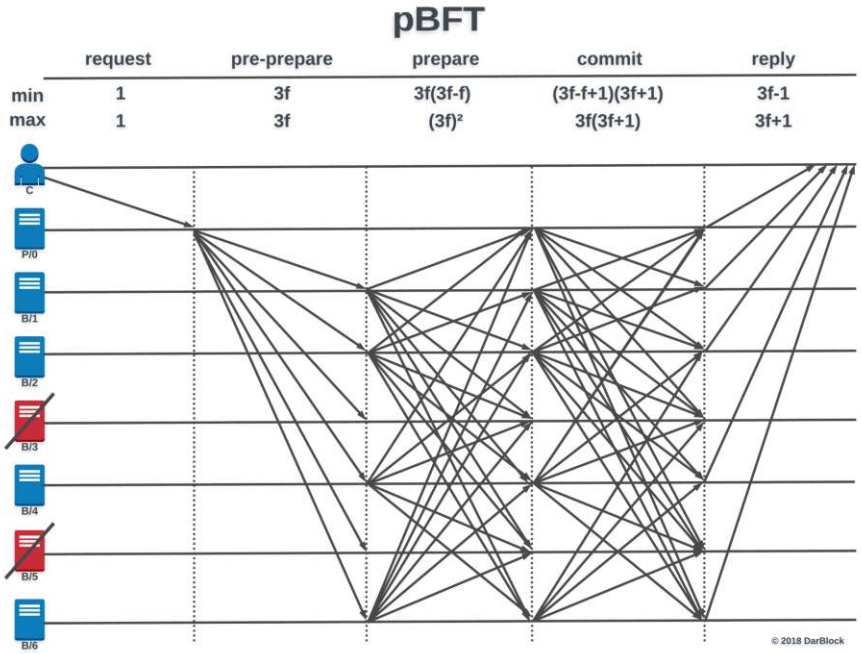

Figure 4.2: PBFT message exchange [PBF].

Regarding the performance, the PBFT requires each node to communicate with each

other node, which can generate high network traffic, thus requiring extremely high performance from the network (see Fig. 4.2). The identity of each node is known, so no anonymity is present in the system. In Ripple, the round consensus is achieved in few seconds, which is suitable for a payment scenario. [ZL19].

Regarding scalability, the Ripple performs better than the PBFT. PBFT, because of the massive message exchange, is suitable for small networks with high network performance. Ripple can be appropriate for large-scale networks, and contrary to the PBFT it has a huge TPS (Transaction per Second) hence, it has good scalability. [ZL19].

Both consensus algorithms require private blockchain because to achieve such techniques, the identity of all the nodes needs to be known. However, on the cost of decentralization, strong consistency and high efficiency are provided hence they are suitable for some commercial and medical scenarios [ZL19].

Considering the comparisons, the first attempt for the implementation was the PBFT consensus algorithm. In the testbed with IoT devices involved, the given amount of messages led to serious problems. The TCP buffers were full, and the exchanged messages were dropped. Above four patients and approximately 20 minutes, the platform did not get any messages from other nodes and, false failure was detected. It led to falsely excluded nodes, and all the nodes started to operate individually. Therefore, the first implementation was not successful according to the performance, and a new, simpler consensus algorithm was introduced.

The bully algorithm is not a typical blockchain algorithm, but rather a technique used in distributed systems to elect master and slave nodes. Since the bully algorithm is used only if the master node gets unavailable, the only message exchange between edge servers happens, when a new block is broadcasted. Keeping the previous experience in mind, the goal was to achieve minimal interaction between the edge servers, to increase the performance to an acceptable level. That's the reason behind choosing the bully algorithm as the final consensus algorithm for the platform.

4.4 Communication

Another crucial aspect of the platform that should be discussed is communication. Three different levels of communication need to be considered within this platform, i.e., the communication between end devices and edge servers, the communication among edge servers, and the communication between edge servers and the cloud.

4.4.1 End device to Edge server

For the communication between end devices and edge servers, an MQTT message broker is used. MQTT stands for MQ Telemetry Transport and was invented by Dr. Andy Stanford-Clark of IBM, and Arlen Nipper of Arcom (now Eurotech), in 1999. It is a publish/subscribe, simple, and lightweight messaging protocol designed for constrained

devices and low-bandwidth, high-latency, or unreliable networks. The design principles are to minimize network bandwidth and device resource requirements while also attempting to ensure reliability and some degree of assurance of delivery. These principles also turn out to make the protocol ideal of the emerging “machine-to-machine” (M2M) or “Internet of Things” world of connected devices and for mobile applications where bandwidth and battery power are at a premium. MQTT supports basic end-to-end QoS [MQT]. Depending on how reliably messages should be delivered to their receivers, MQTT distinguishes between three QoS levels. QoS level 0 is the simplest one: it offers a best-effort delivery service, in which messages are delivered either once or not at all to their destination. No re-transmission or acknowledgment is defined. QoS level 1 provides more reliable transport. A message with such QoS level is re-transmitted until they get acknowledged by the receivers. Consequently, QoS level 1 messages are certain to arrive, but they may arrive multiple times at the destination because of the re-transmissions. The highest QoS level is QoS level 2. It ensures the unique reception of the message without any duplicates. It is up to the application to select the appropriate QoS level for its publications and subscriptions [HTSC08].

In our example Eclipse Mosquitto [Ecl] is used as a message broker server. For producing messages, the python-paho package [Pyt] is utilized. The QoS level of the broadcasting is 2 because this ensures that each message is delivered only once. It is necessary not to deal with the same job several times, which increases the overhead of the servers. Each job is sent to the same channel to provide the flexibility of subscribing only to one channel and receiving all the produced jobs there.

4.4.2 Communication between edge servers

The edge servers form a P2P network. For asynchronous communication, ZeroMQ is used. As per the documentation:" ZeroMQ (also known as ØMQ, 0MQ, or zmq) looks like an embeddable networking library but acts like a concurrency framework. It gives you sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast. You can connect sockets N-to-N with patterns like fan-out, pub-sub, task distribution, and request-reply. It’s fast enough to be the fabric for clustered products. Its asynchronous I/O model gives you scalable multicore applications built as asynchronous message-processing tasks. It has a score of language APIs and runs on most operating systems." [Zer]. It provides intelligent sockets that can be used for fast, reliable, asynchronous inter-node communication. It also has good community support and a good library for python. Therefore, it was chosen as the library in this thesis.

From the various types of sockets provided, the ROUTER and the DEALER are used. The former for distributing messages, the latter to connect to the ROUTER sockets and receive messages. Synchronous communication happens through HTTP requests between the edge servers. If the master node gets unavailable, the consensus algorithm is initiated through an HTTP request, such as all the other message exchanges regarding the consensus.

4.4.3 Edge server to Cloud

The communication between Edge servers and Cloud happens through synchronous HTTP requests. The edge server can offload the built blockchain to the Cloud via HTTP requests. If the Cloud refined the CNN model, it informs the edge server via an HTTP request that the new model is available for download.

4.4.4 Security

Another aspect that needs to be tackled considering communication is security. Two main parts of the system exist, where security is considered and handled. The first level is the communication between edge devices and the end devices. The end devices, that are simulating monitoring devices attached to patients and providing sensor data. It is broadcasted to the edge devices via the MQTT message broker. To keep the blockchain private, only authorized edge servers can be registered to the MQTT message broker therefore only authorized edge servers can receive the data coming from end devices. That measurement ensures that no malicious third party catches the sensitive data.

The second layer, where security is considered is the communication between the edge servers and the cloud. The edge server is offloading the blockchains in a certain period to the cloud for further examination and deeper analysis. HTTP as a communication protocol is used, but the data sent in the request body is encrypted. The secret key to decode the data is only known by the edge servers that encode the messages and the cloud that decodes them. In that way, no malicious third party can observe the blockchain containing sensitive information unless they possess the secret key.

4.5 Implementation details

The whole platform is implemented in Python3 language. The implementation consists of three main modules: cloud, edge server, and end-devices. Additional three modules are introduced: (1) evaluator responsible for evaluating the blockchains, (2) statistics responsible for providing statistical information, and (3) publish a new server module responsible for distributing new edge server information among the edge devices.

4.5.1 Data pre-processing

First of all, the test data needs to be split to have proper subsets of the data. It can be done by the end device. The end device must run with the parameter *–preprocess*, where the parameter must define the test data file in *.csv* format. This data is not provided within this thesis, it must be downloaded from *kaggle.com*. Once the data is downloaded, the script will read the file specified in the arguments. Because the heartbeats (rows) are grouped by categories, the script must reshuffle the rows in a random manner to have more diverse data by patients. After reshuffling, the script reads the data in a 1209-row window, and each of them is written into *.csv* files named with the patient prefix and the numerical order from 1 to 18, such *patient_1.csv*, *patient_2.csv*, etc. At the end, in

the *patient_test_data directory* eighteen *csv* files representing patients each with 1209 heartbeat information are waiting to be broadcasted.

4.5.2 Blockchain

The blockchain module, as the name implies is responsible to build the blockchain ledger. First, the structure of the two main building blocks, namely the block and the blockchain is discussed.

The block is the fundamental building block of the blockchain. It has the following attributes:

- **index** - refers to the index of the block in the chain where it is chained in
- **timestamp** - the time when the block was created
- **previous_hash** - the hash of the previous block in the chain
- **hash** - the hash of the block. The whole block object is converted into a json string, from which with the help of *sha256* python library a hash is computed
- **signalId** - the id of the heartbeat signal
- **transactions** - the samples, that identify the heartbeat
- **proposer** - the id of the node, that proposed the block
- **arrhytmia_category** - the arrhytmia category predicted by the system based on the samples
- **job_created** - the timestamp, when the job was initiated. It is equal to to the *created* attribute of the incoming heartbeat

The blocks are the basic building elements of the chain. Each block is chained to another block, based on the previous hash, that is contained in each block (Figure 4.3). With that structure, to tamper a block in the chain, the whole chain should be replaced. This ensures the immutability of the chain. Once a block is created and added to the chain, the alteration is not possible anymore. The first block of the chain is the genesis block. It is a special block generated by each chain individually, containing no useful information.

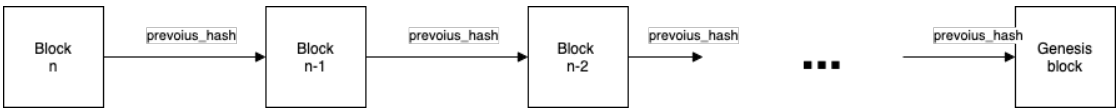

Figure 4.3: Chaining blocks into blockchain.

The blockchain object has the following attributes:

- **chain** - the actual chain containing initially the genesisis block, and the added blocks

To operate on blockchains *BlockchainService* class provides us with a helping hand. It is a singleton class, which can return the requested blockchain if exist, or create a new one with the given patient id. This class is responsible for adding new blocks to the blockchain of the given patient as well. The blockchain module contains converters to convert the JSON objects (blocks, blockchains) into a proper format.

It is worth mentioning that one classic blockchain concept is missing here. The majority of blockchains contain a transaction pool. It is a pool fed by transactions that need to be involved in the next block. The initial implementation of the platform did not send whole heartbeats, but the samples every eight milliseconds. This data was the transaction. When the block creation was commenced all the samples were red from the pool to construct the block. As one can predict, this approach led to serious performance and scalability problems, later on, so the part was refactored and, since a job queue already existed in the system, the transaction pool lost its purpose and was entirely omitted.

4.5.3 Scheduler

The scheduler module takes care of the job queue, scheduling and starting the jobs, and predicting the arrhythmia category.

The *CNNPredictor* as the name would suggest, is responsible for predicting the arrhythmia category via the CNN Keras model. It is responsible for loading the model that is stored in *h5* format. If it is not stored locally yet, the predictor downloads the latest version from the cloud. The model can be refreshed as well, that is a download followed by a reload of the model. The core functionality of the predictor is to predict the arrhythmia categories. It takes one heartbeat and creates the probability vector for it based on the loaded model. Once the vector is created, the category with the highest likelihood is returned.

The *QueueManager* is responsible to manage the job queue that is populated by the incoming heartbeats from the end devices. It can add the new incoming job to the queue. After adding a new job, it is sorted by the *created* attribute of the job. It ensures that each time the oldest job is on the top of the queue. It is also responsible to mark a job if it is executed. In this case, the job is dropped from the queue.

JobScheduler as the name implies takes care of scheduling and executing the jobs. When the first job appears on the channel, the scheduler is initiated, starts to populate the *QueueManager*, and a new Thread is created to perform the execution of the jobs. To start a job, an eternal while loop is initiated. It checks whether the current node should execute the job. It is possible if no ongoing election is present in the network, and the current node is elected as a leader. In that case, the lock is acquired, and the next job is taken from the queue, which is executed and broadcasted to the other nodes.

The *TaskSolver* is responsible for creating the blockchain. Once it is called, it creates the new block, predicts the arrhythmia. It adds the block to the chain and refreshes the leader's heartbeat information. Afterward, a new message is broadcasted among the peers with the new block. The whole process is presented on Figure 4.4.

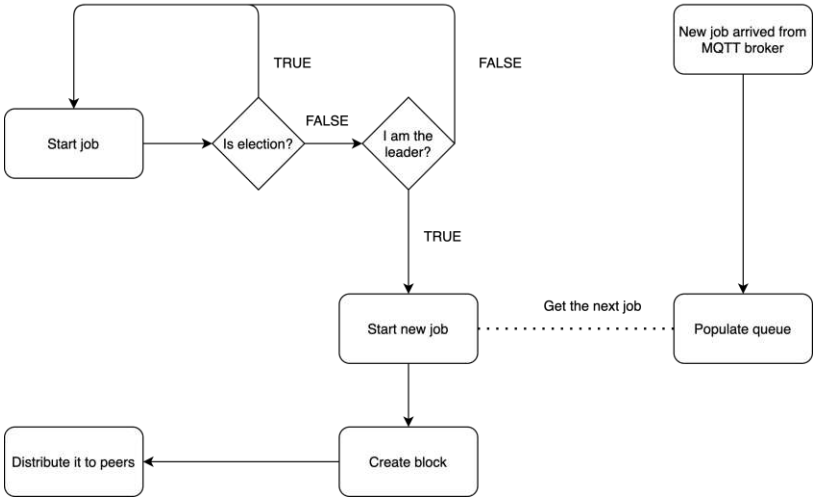

Figure 4.4: Block creation process.

4.5.4 CNN model

The core functionality of the cloud is to train and refine a Keras model. The instructions of [Kag] are followed to build an appropriate model. This model has promising results on the dataset, that is used by also our thesis.

At the first time starting the cloud a new model needs to be initialized. The following steps describe how a new model is initialized:

1. The data needs to be loaded. The training dataset (mitbih_train.csv) is loaded via pandas in *csv* format.

2. The *TrainingsetService* prepares the data for the training. Because the dataset is highly unbalanced and some arrhythmia categories are overrepresented, the set is resampled with 20 000 samples. Therefore, after the resampling is done, all the arrhythmia categories are represented equally by 20000 samples. If a category has more than 20000 samples, then it is simply reduced to that amount.

3. The last column of the resampled dataset contains the category. The five different values there form the five arrhythmia categories.

4. The model is constructed with 3 one dimensional convolutional layers (each sample is represented by a one-dimensional array) and three fully connected layers. The

activation function for each layer is *relu*. At the end of the fully connected layers, a softmax function is used, that provides a probability vector representing each category with a probability. The one, with the highest probability, is the best prediction of the model.

5. The model is trained with the prepared and upsampled training set. The adam optimizer is used with 40 epochs. Each epoch takes approximately 64 seconds, so the whole training is around 40 minutes.

6. After the model is trained it is saved in *.h5* format. This model can be downloaded and used by the edge servers.

As we stated earlier, the model can be refined as well. In that case, the samples coming from edge servers are used. The procedure is the same, except that the already created model is loaded at the beginning, and the blockchain data is used to refine the model. However, if the blockchain data is not diverse enough (there aren't enough categories represented), the model is not refined. If a model were trained with data that has overrepresented categories, it would falsify the predictions with the new model.

4.5.5 Consensus algorithm

Within the consensus algorithm, as was mentioned earlier the bully algorithm is implemented. The bully algorithm is based on the leader election, where the leader is responsible for executing the jobs in our case. To set a new leader, an election should be initiated. An election is initiated if a new edge server is added to the peers, or five seconds passed during which no message was exchanged between the elected node and the others.

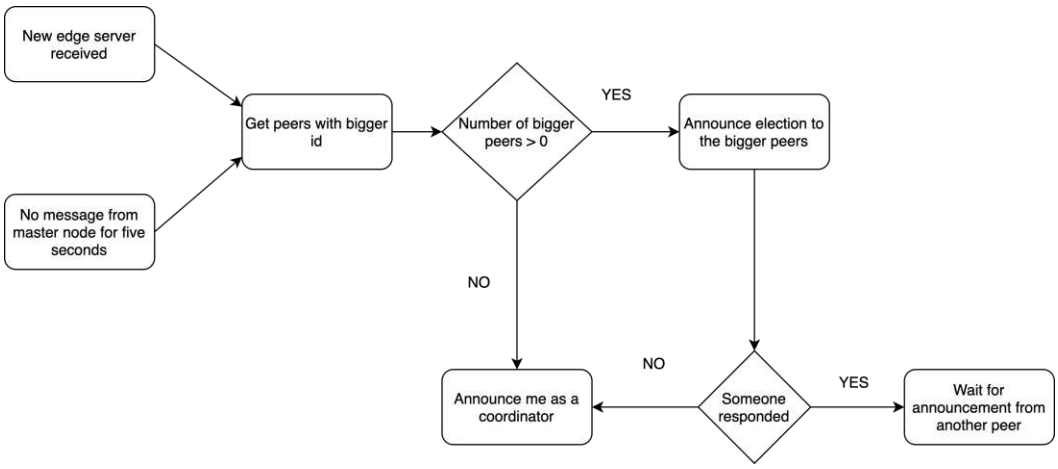

Figure 4.5: Election flow chart.

To initiate the election, a peer must announce one. The *ElectionService* is responsible for that. If an election is initiated, the election state is set to prevent any job execution

during the ongoing election. The peer fetches all the other peers, that have a bigger id, than the actual peer. In this case, bigger means a comparison of ids in alphabetical order. If bigger peers exist, the election is announced through an HTTP request via *AnnouncerService*. The synchronous request must be justified by the peer with a bigger id, i.e., an HTTP 200 must be sent as a response. If a peer with a bigger id responds to that it means, it is active, and the actual peer should wait until a coordinator is announced. If none of the peers responded, the actual peer is the peer that is alive and has the biggest id. In this case, the actual peer sends an announce coordinator HTTP message to the other peers with smaller ids via *AnnouncerService*. The whole scenario is visible on Figure 4.5.

CHAPTER 5

Evaluation methodology

In this chapter, an overall description of the evaluation methodology is provided. The basic approach describes one test case in general, and the measurements are discussed as well. The four aspects of the platform are described, e.g., the performance, fault tolerance, edge scalability, reliability. At last, the test environment is provided in which the tests are conducted.

5.1 Basic approach

The evaluation method informs how the platform is tested, what aspects of the platform are analyzed and evaluated. However, first, the basic approach is discussed.

The extracted data set by Kachuee et al. [KFS18] consists of two types of data set, one for training the models and one for the testing. The testing data set contains 21 892 heartbeats. One heartbeat contains 187 samples, which at 125 Hz sampling rate means the 1.496-second duration for one heartbeat signal. As the MIT-BIH Arrhythmia database uses a half-hour time slot to collect the heartbeats, the thesis will follow this policy as well. At this amount of heartbeats contained in the testing data set appr. eighteen subsets can be generated, each containing heartbeats worth for half-hour duration. Before the heartbeats are divided into 18 subsets, a random shuffling is performed.

Therefore one test case takes a half-hour to be finished. To examine the different aspects of the platform, the tests have various numbers of patients, edge servers, and faulty edge servers.

5.2 Measurements

The different measurements that are taken into account to examine the platform are visible in the table 5.1. In each scenario, the described performance measurements are

Name	Abbr.	Description	Unit
Accurate predictions	Acc.	Measures the prediction accuracy	%
Min proc. time	Min.	Min proc. time of one job	sec
Max proc. time	Max.	Max proc. time of one job	sec
Avg proc. time	Avg..	Avg proc. time of one job	sec
Sample lost	Smp. l.	Heartbeats lost during the scenario	%
Undecided predictions	Un. pr.	Heartbeats unable to be predicted	%
Proposer rate	Prop. r.	Rate of proposed blocks per proposer	%
Blockchains equality	Eq	Blockchain equality among the devices	Boolean

Table 5.1: Measurements to evaluate performance.

measured and evaluated.

5.3 Evaluation process

To evaluate the results, and evaluation script is written. This script takes all the blockchains in JSON format generated by the different use cases, and evaluates them in the following manner:

1. The script takes two command-line arguments. The first one is the root directory, where the generated blockchains are found for all the nodes. The second one is the root directory of the test data in *.csv* format because it provides the proper arrhythmia categories that can measure the accuracy of the CNN network.
2. First, the blockchains are evaluated, whether they are identic, or not. It is done by taking the md5 hash of the content of the JSON files and comparing them to each other. If all the hashes are the same, that means that all the file contents are the same. In this case, the script continues. In case, if there is a blockchain that is different from the others, the script gives an error message and terminates. It makes only sense to evaluate the performance if the blockchains are the same. Otherwise, the whole concept of the platform is violated.
3. If all the blockchains are the same, the first node is taken as a reference. The script iterates through all the patients (all the *.json* files), evaluates the measurements seen in Table 5.1 on them, and provides the results for each patient individually.
4. After the evaluation is done the results are printed on the screen and are available for further examination

5.4 Aspects

The four aspects of the platform are presented, that need to be evaluated based on different test setups and scenarios.

5.4.1 Performance analysis

The first aspect of the system is performance. To check the performance the test cases are conducted with different amounts of patients. All the other actors stay the same during every test. As it was discussed earlier the performance can be examined for up to 18 patients. Therefore seven use cases are constructed that are visible in table 5.2. These use cases are run and, for each use case, the measurements in Table 5.1 evaluated.

ID	Number of patiens
UC_1	1 patient
UC_2	5 patients
UC_3	7 patients
UC_4	10 patients
UC_5	13 patients
UC_6	16 patients
UC_7	18 patients

Table 5.2: Summary of use case scenarios constructed for the thesis.

The other actors of the system stay the same in each scenario. That means:

- **3** edge servers
- **1** cloud
- **1** message broker

5.4.2 Fault tolerance

The fault tolerance tests should discover the performance of the platform if faulty nodes are present in the system. Within the fault tolerance tests, all the use cases from Table 5.2 are run again. However, this time after approximately 10 minutes one of the servers is shut down, and after 20 minutes another one is killed. It implies that the system runs 10 minutes without failure, 10 minutes with one failed node, and 10 minutes with two failed nodes, hence with only one node alive.

The shut down is done on the master node in the system, which actively calculates and distributes blocks in the platform. It is necessary to see how the failure affects the performance. If not the master node would be shut down the effect would be invisible. The system detects the fault if the active master node does not distribute the blocks. A new election can be initiated only if the master node gets unavailable.

The performance results of the scenarios under this chapter are interesting in light of the results of the previous chapter. A comparison needs to be done to see how the performance was affected by the failure.

It is also worth mentioning, that if any difficulties are detected in the performance analysis scenarios, a new upper limit of the platform is identified, i.e., a smaller amount of patients. In that case, not all the fault tolerance tests are run again only up to the new upper limit of the number of patients. The number of the other actors of the system stays the same in each scenario:

- **3** edge servers (at the beginning)
- **1** cloud
- **1** message broker

5.4.3 Edge scalability

The edge scalability checks, what is the reaction of the platform at the various amount of edge servers. Because of the limited physical resources available for us, the maximum amount of edge servers is four. So in this scenario, four tests are run, at first with one edge server, and afterward, the servers are increased by one in each scenario. The number of patients is determined by the performance analysis. The upper limit of the platform gives the number of patients in this scenario. The number of the other actors stays the same. The same measurements are evaluated, and each scenario is evaluated in the light of the other scenarios. The result of the test shows how the number of edge servers impacts the measurements. Hence the actors:

- **n** edge servers (where n is between 1 and 4)
- **1** cloud
- **1** message broker
- **n** patients - the upper limit of the platform regarding the number of patients

5.4.4 Reliability

The reliability tests show the range within one test case can vary by running it at different times. In this case, one test with the same setup is conducted ten times in a row. The ten results are compared, and the range is checked in which the results are spread. To visualize the results box plot charts are used. The actors in this scenario are:

- **3** edge servers
- **1** cloud
- **1** message broker
- **n** patients - the upper limit of the platform regarding the number of patients

The measurements are the same, but under reliability tests also new measurements are introduced. The new characteristics with description can be seen at Table 5.3. To generate the values a web tool is used BoxPlotR. [Box]

Name	Description
Upper whisker	The maximum value of the given measurement
3rd quartile	The value under which lie the 75% of the values
Median	The value that separates the higher half from the lower half
1st quartile	The value under which lie the 25% of the values
Lower whisker	The minimum value of the given measurement
Mean	The average value of the numbers

Table 5.3: Box plot statistics measurements.

5.5 Test setup

In this section, a quick introduction to the setup of the tests is given. As per the architecture, four main actors participate in all the scenarios. The difference is only in the amount of them. The five actors are cloud, edge server, message broker, patients.

Ideally, all the actors should be simulated on different devices. This means, each patient's end device should be a different device, all the edge servers should have a dedicated computer. The message broker should be located on a different device and, in the end, the cloud needs to be located relatively far away from the setup, simulated by a powerful machine. With the biggest use case scenario, this would mean 23 different devices. Unfortunately, due to a lack of resources, this realistic setup can't be simulated. The available resources allow me to use the following setup:

- Each edge device is simulated on a different raspberry pi
- the message broker is simulated on a raspberry pi
- the cloud is simulated on my local computer
- the patients are simulated on my local computer

5.5.1 Local computer

Our local computer has the following specification:

- 2.3GHz dual-core Intel Core i5, Turbo Boost up to 3.6GHz, with 64MB of eDRAM
- 8GB of 2133MHz LPDDR3 onboard memory
- 256GB SSD

The local computer's properties are good enough to simulate a cloud on it. Admittedly, for the patients would be good to have at least another powerful machine, but except for the heat generated by the local machine, no serious issue was caused by simulating such a high amount of actors on one computer.

5.5.2 Raspberry Pi

All the Raspberry Pi's occurring in the testbed was Raspberry Pi 3 B models, with the following specifications:

- Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
- 1GB RAM
- BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
- 16 GB flash drive

The Raspberry Pi 3B model fulfills the requirements of an IoT edge device. However, simulating one edge server on one device was enough pressure to take regarding the performance. Out of curiosity, simulating more edge servers on one device led to serious performance issues and crashes. Probably because the TensorFlow 2.2 is not well supported on 32-bit architecture, which is the recommended OS architecture used by the Pi. Using 64-bit OS on them caused a serious slow down even without starting an edge server. Furthermore, simulating patients on the RPi caused crashes on the devices when more than two patients' data on one machine was simultaneous.

The edge devices and MQTT broker (represented by Raspberry Pis) were connected wired, to speed up the P2P communication between them. The connection to the local computer (cloud and patients) was wireless.

CHAPTER 6

Evaluation results

This chapter provides the results from the conducted tests as described previously in Chapter 5.

6.1 Performance analysis

In the performance analysis, the use cases with the growing patient numbers are analyzed. After seven patients, the platform has shown difficulties that made the platform inadequate to deploy it in such a mission-critical environment. Hence, the upper limit of the platform is defined with seven patients.

6.1.1 Use case 1

The evaluation returned by the script is seen on Table 6.1. The platform's performance on one patient is promising. No information is lost, all the samples are delivered, no undecided predictions, and the total amount of signals is processed. The percentage of accurate arrhythmia predictions is above 96 %, which is considered good as well. The workload is concentrated at one proposer which is the expected behavior because no failure was present during the test. The average processing time is 30 milliseconds, and the max processing time is around 1 second. It is worth mentioning, that every 1.496 seconds a new job is generated, so even the worst processing time did not exceed that.

	Acc.	**Min.**	**Max.**	**Avg.**	**Smp. l.**	**Un. pr.**	**Prop. r.**
P1	96.27%	0.030 s	1.004 s	0.057 s	0%	0%	p1: 100% p2: 0% p3 0%

Table 6.1: Use case 1 evaluation - 1 patient.

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.	Prop. r.
P1	96.27%	0.017 s	12.969 s	0.827 s	0%	0%	p1: 100% p2: 0% p3: 0%
P2	95.53%	0.021 s	12.757 s	0.845 s	0%	0%	p1: 100% p2: 0% p3: 0%
P3	95.20%	0.025 s	11.956 s	0.880 s	0%	0%	p1: 100% p2: 0% p3: 0%
P4	96.52%	0.033 s	12.598 s	1.010 s	0%	0%	p1: 100% p2: 0% p3: 0%
P5	95.28%	0.018 s	12.651 s	0.790 s	0%	0%	p1: 100% p2: 0% p3: 0%

Table 6.2: Use case 2 evaluation - 5 patients.

6.1.2 Use case 2

The evaluation returned by the script is seen on Table 6.2. The platform's performance on five patients is promising as well. No information is lost, all the samples are delivered, no undecided predictions, and the total amount of signals is processed. The percentage of accurate arrhythmia predictions is around 95 % by each patient, which is considered good as well. The workload is concentrated at one proposer which is the expected behavior because no failure was present during the test. The worst average processing time is 1 second, and the max processing time is around 12 seconds. In the light of the fact that every 1.496 seconds a new job is generated, it is visible that some of the jobs were queued.

6.1.3 Use case 3

The evaluation returned by the script is seen on Table 6.3. The platform's performance on seven patients shows some difficulties. One heartbeat is lost in one patient, which is good at such a level of pressure. No undecided predictions and the total amount of received signals is processed. The percentage of accurate arrhythmia predictions is around 95 % by each patient, which is considered good as well. The workload is concentrated at one proposer which is the expected behavior because no failure was present during the test. The worst average processing time is around 80 seconds, and the max processing time is around 170 seconds. In the light of the fact that every 1.496 seconds new job is generated, it is visible that some of the jobs were queued and processed after the patients finished producing samples. In this case, it is almost 3 minutes (10% of the whole experiment time) after the generation of the heartbeats is over.

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.	Prop. r.
P1	96.27%	0.050 s	169.947 s	79.179 s	0%	0%	p1: 100% p2: 0% p3: 0%
P2	95.53%	2.508 s	169.932 s	79.047 s	0%	0%	p1: 100% p2: 0% p3 0%
P3	95.19%	1.765 s	169.716 s	78.900 s	0.08%	0%	p1: 100% p2: 0% p3 0%
P4	96.52%	2.328 s	170.144 s	79.058 s	0%	0%	p1: 100% p2: 0% p3 0%
P5	95.28%	2.109 s	169.293 s	78.900 s	0%	0%	p1: 100% p2: 0% p3 0%
P6	96.85%	2.135 s	169.896 s	79.241 s	0%	0%	p1: 100% p2: 0% p3 0%
P7	95.94%	2.296 s	169.694 s	79.257 s	0%	0%	p1: 100% p2: 0% p3 0%

Table 6.3: Use case 3 evaluation - 7 patients.

6.1.4 Rest of the use cases

After seven patients the performance of the platform showed difficulties to an extent, that is unacceptable for a mission-critical system. Hence, this section only provides the measurements of the system depicted by charts.

The first chart 6.1 depicts the accurate predictions according to the number of patients. Although a slight decrease is present after increasing the number of patients, the platform regarding the accurate predictions performs very well. The accuracy is around 96% and, only minor differences could be spotted.

The second chart 6.2 shows the biggest problem of the platform. It is visible that after seven patients, the increased pressure to handle the requests to build the blocks leads to massive losses regarding the heartbeats. The problem is in the communication. In the use case, it was visible, that all the heartbeats were sent towards the edge servers. However, on the other side, the logs show that not all the messages have arrived. Although the TCP buffer was relieved by applying the bully algorithm, the massive message exchange at a high pressure still led to omitting messages. The messages were not dropped by the application but by the underlying system. At ten patients, the platform loses a third of

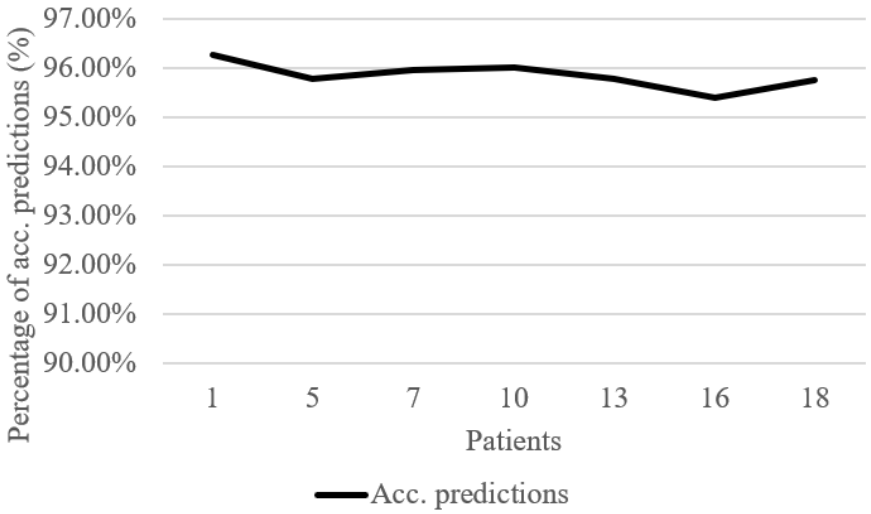

Figure 6.1: Accurate predictions for different amount of patients.

the heartbeats. At the highest amount of patients in the experiment, around two-third of the heartbeats are lost. That ratio is unacceptable for a mission-critical system and that is why the upper limit of the system is defined as up to seven patients.

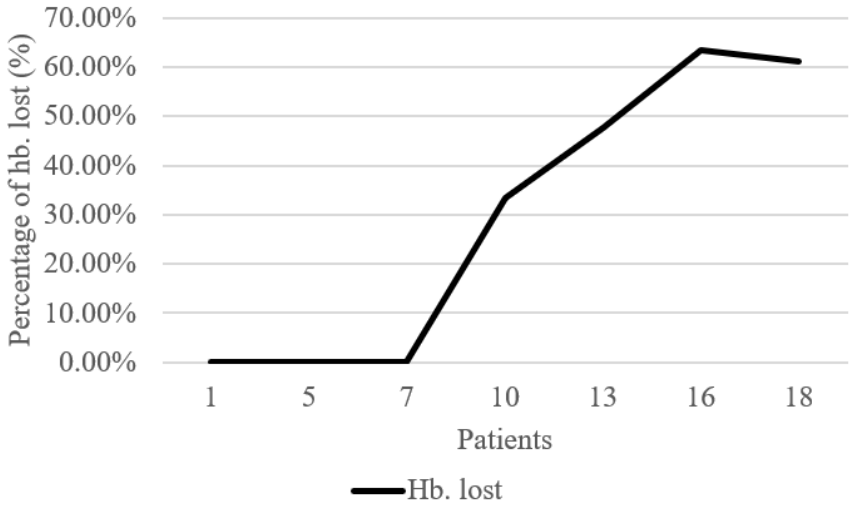

Figure 6.2: Heartbeat loss rate for different amount of patients.

The average processing time seen in Figure 6.3 is the most descriptive above all the measurements regarding the processing time. It depicts, how the average processing time increases after increasing the number of patients, hence increasing the workload. The expectation is that by increasing the workload the average processing time will increase

with linear growth. However, after ten patients, the chart starts to sink. This paradox behavior can be understood by keeping in mind the heartbeat lost ratio. The increased number of heartbeats thrown away by the TCP buffer decreases the workload. That is the cause of the shape of Figure 6.3.

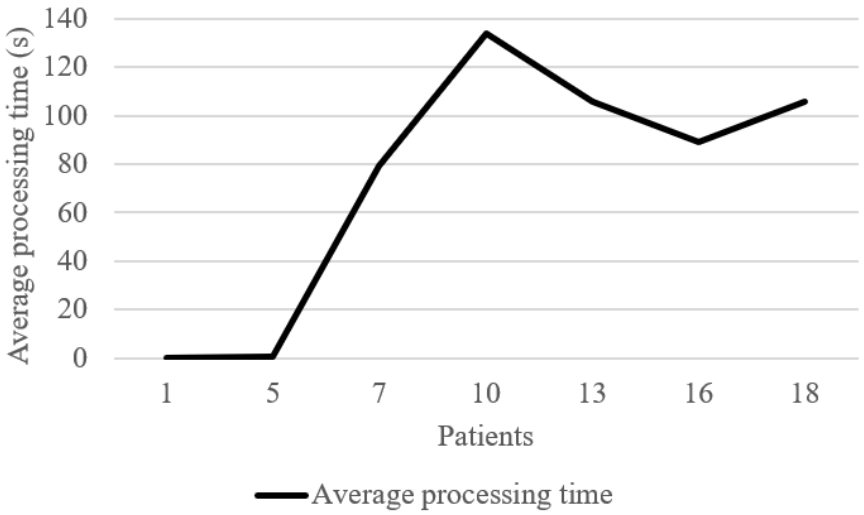

Figure 6.3: Average processing time for different amount of patients.

The max processing time, shown in Figure 6.4 depicts the maximum processing time between a heartbeat was produced and it was added to the block. Because of the heartbeat loss, it shows a similar shape as the average processing time. The heartbeats are processed in sequence, and they are buffered in the queue hence the maximum processing time provides information, how long it takes to process the last generated heartbeat. It is useful because it shows us how much time is needed after the test is finished to see the final results (blockchains).

6.1.5 Summary

In this section, the overall picture of the performance analysis is summarized. The first measurement is accurate predictions. The platform can stick to the success rate predicted by [Kag]. The values are over 94% in each case, and no undecided predictions are present in the blockchain. Therefore, the predictions are reliable, and the platform is useful to predict arrhythmia categories.

The average procession time varies across the different use cases. Until five patients, it could be considered excellent. At five patients, it barely exceeds 1 second. Considering the 1.496 seconds period producing heartbeat samples leads to nearly immediate results. After seven patients, a delay is experienced. That delay, according to the increased number of collected data means in real life that after monitoring seven patients for 30 minutes, 80 seconds needs to pass to get the final state of the blockchain. The average

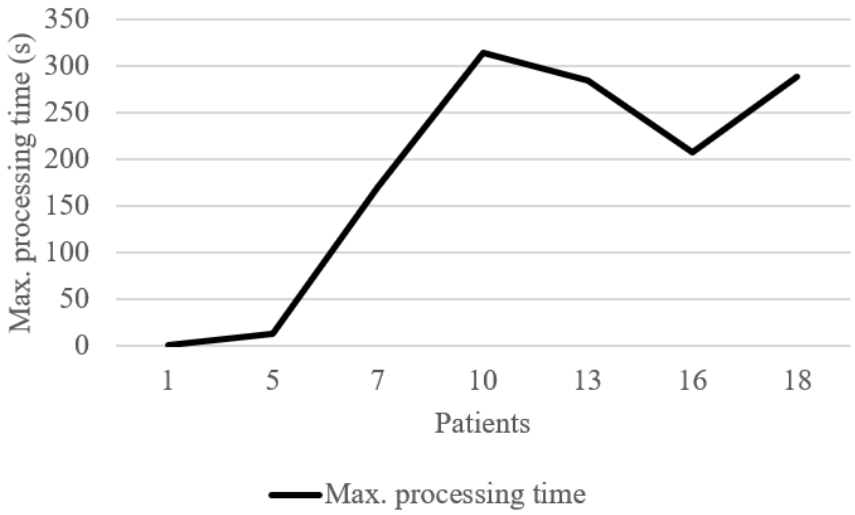

Figure 6.4: Max processing time for different amount of patients.

processing time at ten patients and 18 patients is around 2 minutes, but this is because of the high number of lost heartbeats.

The max and the min processing time are various by different use cases, as we expect from edge values. However, the particular values across the patient don't vary significantly. This means in a real-life scenario, that all the patients have appr. the same values, hence the platform is balanced and, no starving of patients is experienced.

The sample lost shows the greatest lack of the platform. After seven patients, the number of the lost heartbeats has reached an unacceptable level and undermines the reliability. In a conclusion, the platform scales well to a particular threshold, but afterwards too much information is lost.

6.2 Fault tolerance

As it was discussed earlier, the upper limit of the platform is seven patients in parallel. Therefore, the fault tolerance tests are presented and discussed only for up to seven patients, e.g., the first three use cases. The values are analyzed in the light of the previous scenarios, where no faulty nodes were injected during the tests.

6.2.1 Use case 1

The evaluation returned by the script is seen on Table 6.4. The fault tolerance is promising in the scenario of 1 patient. The max processing time is the only main difference, where the idle time is included, that was necessary to detect the failure of the master node. The average processing time is 0.1 seconds, which is considered good, however, it is doubled

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.	Prop. r.
P1	96.27%	0.017 s	8.365 s	0.100 s	0%	0%	p1: 39.86% p2: 33.66% p3: 26.47%

Table 6.4: Use case 1 evaluation - 1 patient (fault tolerance).

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.	Prop. r.
P1	96.27%	0.032 s	14.668 s	1.046 s	0%	0%	p1: 32.50% p2: 33.25% p3: 34.25%
P2	95.53%	0.036 s	15.314 s	1.122 s	0%	0%	p1: 32.50% p2: 33.25% p3: 34.25%
P3	95.20%	0.029 s	14.867 s	1.079 s	0%	0%	p1: 32.50% p2: 33.25% p3: 34.25%
P4	96.52%	0.029 s	15.949 s	1.054 s	0%	0%	p1: 32.50% p2: 33.25% p3: 34.25%
P5	95.28%	0.023 s	599.190 s	1.535 s	0%	0%	p1: 32.50% p2: 33.25% p3: 34.25%

Table 6.5: Use case 2 evaluation - 5 patients (fault tolerance).

than at the non-fault tolerance scenario. The workload distribution is also considered balanced.

6.2.2 Use case 2

The evaluation returned by the script is seen on Table 6.5. The fault tolerance is promising in the scenario of 5 patients. The max processing time is the only main difference, where the idle time is included, that was necessary to detect the failure of the master node. The average processing time is 1.5 seconds, which is considered good, however, it is 50% higher than at the non-fault tolerance scenario. The workload distribution is also considered balanced.

6.2.3 Use case 3

The evaluation returned by the script is seen on Table 6.6. The fault tolerance is promising in the scenario of 7 patients. The max processing time and the average processing time show very little differences with the non-fault tolerance scenario. This proximity to the previous results comes from the fact that the jobs needed to be queued anyway, so

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.	Prop. r.
P1	96.27%	0.769 s	176.052 s	81.992 s	0%	0%	p1: 30.52% p2: 30.10% p3: 39.38%
P2	95.53%	0.576 s	176.338 s	81.983 s	0%	0%	p1: 30.52% p2: 30.10% p3: 39.38%
P3	95.20%	0.067 s	176.235 s	81.586 s	0%	0%	p1: 30.52% p2: 30.19% p3: 39.29%
P4	96.52%	0.222 s	175.684 s	81.660 s	0%	0%	p1: 30.52% p2: 30.19% p3: 39.29%
P5	95.28%	0.410 s	176.544 s	81.724 s	0%	0%	p1: 30.52% p2: 30.19% p3: 39.29%
P6	96.85%	0.054 s	176.383 s	80.772 s	0%	0%	p1: 31.18% p2: 30.19% p3: 38.63%
P7	95.94%	0.061 s	176.417 s	80.780 s	0%	0%	p1: 31.18% p2: 30.19% p3: 38.63%

Table 6.6: Use case 3 evaluation - 7 patients (fault tolerance).

the idle time does not affect significantly the results. The workload distribution is also considered balanced.

6.2.4 Summary

In this section, the results of the fault tolerance scenarios are discussed, and an overall picture is presented. If we look at the results, two important conclusions can be made. The fault tolerance scenarios, in terms of accurate predictions or sample, lost performed not worse than the original use cases. From the values that can be derived, that the platform overall can handle the faulty scenarios and no significant performance issue is presented. However, this does not change the fact that after seven patients the platform reaches its limits.

6.3 Edge scalability

The edge scalability is evaluated based on the average values of the seven patients in one scenario. The interesting measurements are taken for each scenario, and the average

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.
1 Edge server	95.94 %	2.21 s	165.25 s	50.61 s	0.0 %	0.0 %
2 Edge server	95.94 %	2.29 s	159.21 s	66.57 s	0.0 %	0.0 %
3 Edge server	95.94 %	1.88 s	169.80 s	79.08 s	0.0 %	0.0 %
4 Edge server	95.94 %	2.06 s	206.48 s	75.04 s	0.0 %	0.0 %

Table 6.7: Edge scalability - overall picture.

value over the seven patients is examined. This helps us to identify the characteristics of the platform on different amounts of edge servers without the unnecessary details.

The results are visible on Table 6.7. Each measurement is analyzed by comparing it to the other scenarios.

The first measurement is the accuracy of the CNN model. It is visible that by rounding all the values are the same. It implies that only minor differences are present between the scenarios, which is the expected behavior. The edge server is working with the same model hence, no impact is taken by increasing the number of the edge devices. The minimum and maximum processing time results serve as an interesting fact rather than values from which a strong conclusion could be made. As edge values, they vary across the devices without a significant pattern.

The average processing time provides us more interesting results. It is visible that by increasing the number of edge servers, the average processing time also increases. However, the growth is not linear in terms of edge servers. It depends probably on the TCP communication, where at one edge server scenario no block needs to be distributed and, at four edge server scenario, the master node needs to send the calculated block to three more edge servers. In general, we can say that the performance is better with fewer edge servers, but this sacrifice brings the drawback of the decreased number of possibly faulty nodes in the system.

The sample lost and undecided predictions depict, that increasing the number of edge servers does not affect the reliability of the system. On each level, no samples are lost, and no undecided predictions were made.

6.4 Reliability

The reliability measurements are visible in the Table 6.8. In each scenario, seven patients were simulated and, the rows show the average values by each attempt. Each measurement is depicted on box plot charts (except avg. predictions), which makes them easier to evaluate. The average predictions only differ in the third digit after the comma, but the BoxPlotR rounds it up to 2 digits, which leads to 95.95% for each attempt.

Figure 6.5 shows the heartbeat loss and undecided predictions on the box plot. The y axis represents the values in percentage. It is visible that the ten attempts of one test

	Acc.	Min.	Max.	Avg.	Smp. l.	Un. pr.
1.	95.947 %	2.55 s	169.14 s	73.56 s	0.0 %	0.0 %
2.	95.946 %	2.30 s	171.93 s	70.92 s	0.01 %	0.0 %
3.	95.947 %	2.00 s	240.45 s	73.35 s	0.0 %	0.0 %
4.	95.946 %	2.68 s	236.29 s	75.57 s	0.01 %	0.0 %
5.	95.947 %	2.47 s	164.03 s	73.78 s	0.0 %	0.0 %
6.	95.947 %	2.65 s	205.13 s	69.82 s	0.0 %	0.0 %
7.	95.947 %	2.53 s	183.66 s	78.29 s	0.0 %	0.0 %
8.	95.946 %	1.73 s	198.39 s	63.69 s	0.01 %	0.0 %
9.	95.947 %	2.03 s	230.92 s	64.95 s	0.0 %	0.0 %
10.	95.946 %	1.99 s	199.94 s	80.59 s	0.01 %	0.0 %

Table 6.8: Reliability - overall picture.

case show a minor difference in terms of heartbeat loss and undecided predictions. Hence, the results in the other scenarios can be considered reliable.

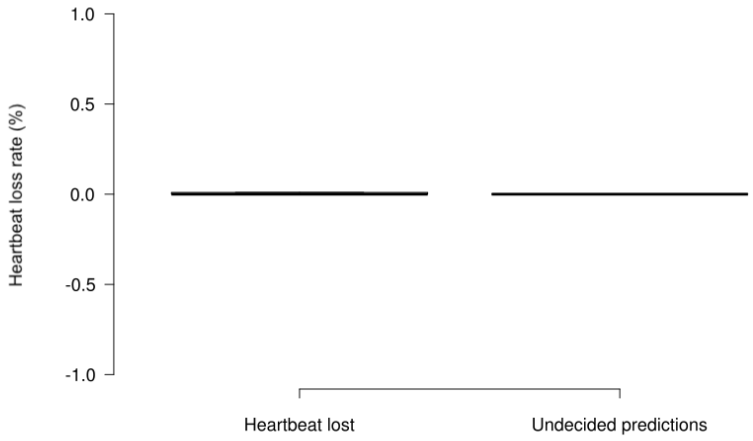

Figure 6.5: Heartbeat loss and undecided predictions.

The minimum processing time (Fig. 6.6) varies between the maximum value of 2.68 seconds and the minimum value of 1.73 seconds. The median is 2.38 seconds, and the mean is 2.29 seconds. The 1st quartile is 2.00 seconds, and the 3rd quartile is 2.55 seconds, which means that half of the values are between this interval.

The average processing time (Fig. 6.7) varies between the maximum value of 80.59 seconds and the minimum value of 63.69 seconds. The median is 73.45 seconds, and the mean is 72.45 seconds. The 1st quartile is 69.82 seconds, and the 3rd quartile is 75.57 seconds, which means that half of the values are between this interval.

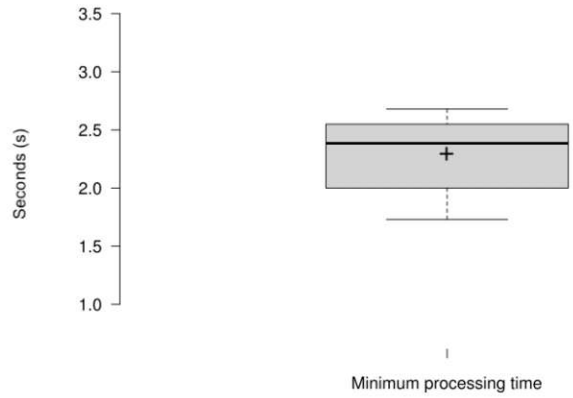

Figure 6.6: Min processing time.

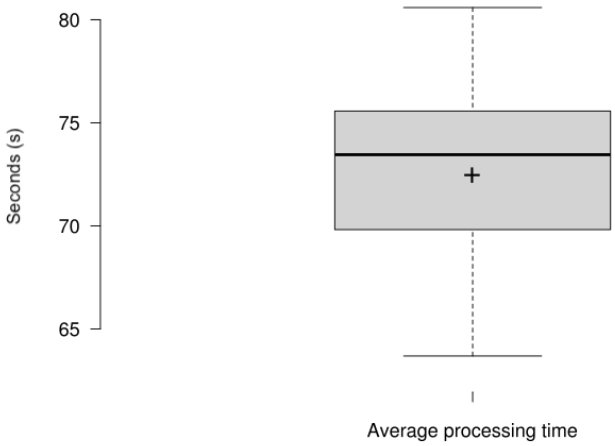

Figure 6.7: Average processing time.

The maximum processing time (Fig. 6.8) varies between the maximum value of 240.45 seconds and the minimum value of 164.03 seconds. The median is 199.16 seconds, and the mean is 199.99 seconds. The 1st quartile is 171.93 seconds, and the 3rd quartile is 230.92 seconds, which means that half of the values are between this interval.

6.4.1 Summary

Table 6.9 sums up all the stated measurements and gives an overall overview of the box plot numbers. From Table 6.9 it is visible that regarding the sample lost and undecided predictions the system is reliable. A minor sample loss was present at seven patients in

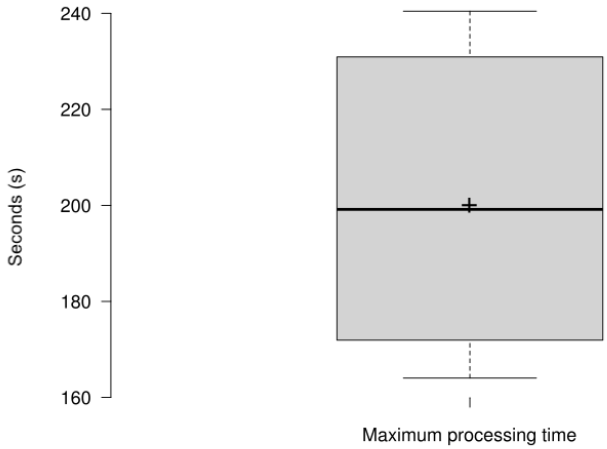

Figure 6.8: Max processing time.

	Min.	Max.	Avg.	Smp. l.	Un. pr.
Upper whisker	2.68 s	240.45 s	80.59 s	0.01 %	0.0 %
3rd quartile	2.55 s	230.92 s	75.57 s	0.01 %	0.0 %
Median	2.38 s	199.16 s	73.45 s	0.0 %	0.0 %
1st quartile	2.00 s	171.93 s	69.82 s	0.0 %	0.0 %
Lower whisker	1.73 s	164.03 s	63.69 s	0.0 %	0.0 %
Mean	2.29 s	199.99 s	72.45 s	0.0 %	0.0 %

Table 6.9: Box plot - overall picture.

general the values are promising. The average processing time shows a higher variation across the attempts. However, in general, 50% of the results vary between 69.82 seconds and 75.57 seconds. The median and mean shows also a minor difference, that is 1 second. The max processing time, as an edge value, shows more variation. However, as an edge value, it can't be used to make solid conclusions. Checking Table 6.8 it is visible that no correlation can be seen between the average processing time and maximum processing time, so the impact of the maximum processing time on the average processing time is not straightforward.

CHAPTER 7

Discussion

In this section, an overall discussion is provided about the thesis. The key findings summarize the results and discuss what can be derived and observed from them. The implications describe the implications of the results and platform in the light of the hypothesis. Limitations analyze what limitations the system has and what the results can not tell us.

7.1 Key findings

The thesis aimed to build a platform that can handle sensor data in a mission-critical system, use AI to make predictions, and create a blockchain ledger out of the evaluated data. The platform was evaluated based on a test plan. It aims to provide results to analyze four aspects of the platform: performance, fault tolerance, edge scalability, reliability. From the results the following indications can be made:

1. The platform performs well up to seven patients in parallel. The data was handled correctly, accurate predictions were made and, no serious information loss was observed.

2. The average processing time at the maximum (seven patients) reasonable pressure is 80 seconds and 95% accuracy. That means after a heartbeat is generated appr. 1 minute and 20 seconds needed to be a wait to receive an arrhythmia category with 95% accuracy

3. The scenarios, where faulty nodes were present in the system, has shown minor difference according to the regular scenarios. That means the platform handles the faulty nodes well.

4. With the rising number of edge servers, the performance is decreasing. However, the number of faulty nodes that can be handled is increasing with each edge server. In our scenario, where four edge servers could be deployed at most, three was determined as an optimal number. For other cases, another number as an optimum can be defined.

5. The reliability tests, where the same test setup was run several times in a row, show minor differences across the different tests. That is visible on the box plot charts, where the range of the results can be examined and analyzed.

All the points meet the previous expectations and it is supported by our hypothesis. The blockchain concept is responsible for the fault tolerance and the introducing overhead is still under an acceptable threshold.

7.2 Implications

The results have been shown that the hypothesis, namely to build a blockchain ledger from the data without introducing significant overhead was correct. Blockchain technology can be deployed on the edge of the network. It increases the fault tolerance of the system without decreasing the performance significantly.

7.3 Limitations

The system was tested with at most four edge servers. We tested our solution on low-powered edge servers to present scenarios as realistic as possible (i.e., Raspberry Pis). The platform was not tested with more than four edge servers; therefore, we don't have an upper limit in terms of edge devices. Just predictions can be made on how the system would react on more edge servers. As previously mentioned, our proposed solution is tested only with RPis, and we did not evaluate our solution with more powerful devices. Hence, within the presented results, one cannot determine whether a significant improvement can be achieved with more computational power.

The results have shown that up to seven patients approximately 80 seconds evaluation time can be observed on average. That means after the half-hour period is done, so much time needs to be counted for each heartbeat on average. It was not tested, whether this value depends on the duration of the tests. The results can not tell whether a one-hour period or a 24-hour period will provide the same average value or increase with time. In terms of the practical use of the system, the most significant limitation is the lost heartbeats. During the message exchange at high pressure (above seven patients), heartbeats were dropped, leading to an unacceptable ratio of data loss. In a mission-critical system, it is crucial to have all the data. This phenomenon defines an upper limit for the system and makes it unusable in practice above seven patients in parallel. However, the results can not tell us whether a network with higher throughput could provide augmentations here.

CHAPTER 8

Conclusion

In this chapter, the research question that was stated at the beginning is answered. The future work and improvement possibilities are discussed and at the end, a summary is provided.

8.1 Research questions

Let's recall and answer the research questions that we introduced in Chapter 1.

(**RQ**1) *What is the maximum amount of patients that the system can handle at once?*

As we stated multiple times earlier, the maximum amount of patients that can be handled in parallel is **seven**. Above that amount, too much information is lost, which violates the usability of such a platform.

(**RQ**2) *How relates the number of edge servers to the fault tolerance and performance?*

The number of edge servers, as from the definition of the consensus algorithm is implied, increases the fault tolerance of the system. The more edge servers are present, the more of them can be faulty. However, the number of edge servers increases also the average processing time (latency), because the more edge servers are present, the more message needs to be exchanged.

(**RQ**3) *What it is the latency-overhead by introducing the blockchain technology compared to the non blockchain solution?*

To answer this question the edge scalability test needs to be considered. The main overhead by introducing blockchain technology comes with the consensus algorithm.

In a test case, where only one edge server is present, no consensus algorithm needs to be involved and no additional message exchange needs to be performed. It is visible, that appr. 30 % increase in the average processing time can be observed after introducing the first edge server.

(**RQ**4) *What is the impact of the faulty nodes on the performance?*

As the fault tolerance test scenarios are visible, minor differences can be observed after introducing faulty nodes in the system. That means the platform performs well in fault tolerance scenarios because the results are not significantly worse than in the scenarios without faulty nodes.

8.2 Future work

In this section, future work regarding the platform is discussed. The results of the system examine each heartbeat individually. It is capable to observe an arrhythmia, but no further analysis or examination is implemented. In the cloud a more complex and further analysis could be to examine the whole set of results, finding patterns or preludes from which complex diseases can be foreseen.

As the main bottleneck of the system is the consensus algorithm, it is worth reconsidering the messaging protocol used among the edge servers. Different messaging protocols can be tested, which can maybe lead to better performance. A more extensive setup can also provide more detailed information about the platform. The lack of resources(physical machines) made us make compromises in the realistic test setup. However, it would be interesting to conduct the test with a higher amount of edge servers, with more powerful cloud, more powerful edge servers, etc.

The scheduling can be reconsidered as well. Currently, if no faulty node is present in the system, all the work is done by only the master node, which was chosen at the start of the test. It was necessary to decrease the message exchange among the edge servers. In the future, this can be reconsidered and improved to get better workload distribution.

8.3 Summary

Within the thesis, we built a platform that predicts arrhythmia from realistic data. The resulting data is collected in a blockchain ledger. The hypothesis at the beginning was correct therefore a lightweight blockchain ledger can improve the fault tolerance of the system without decreasing significantly the performance. All the research questions were answered. Al last the future work and improvement possibilities were discussed.

List of Figures

List of Tables

Bibliography

[AAR+17] 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):174, 2017.

[ACTA17] Orestis Akrivopoulos, Ioannis Chatzigiannakis, Christos Tselios, and Athanasios Antoniou. On the deployment of healthcare applications over fog computing infrastructure. In *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, volume 2, pages 288–293. IEEE, 2017.

[AIH18] Ziti Fariha Mohd Apandi, Ryojun Ikeura, and Soichiro Hayakawa. Arrhythmia detection using mit-bih dataset: A review. In *2018 International Conference on Computational Approach in Smart Systems Design and Applications (ICASSDA)*, pages 1–5. IEEE, 2018.

[Box] Boxplotr: a web-tool for generation of box plots. `http://shiny.chemgrid.org/boxplotr/`. Accessed: 2021-05-11.

[Buc16] Ethan Buchman. *Tendermint: Byzantine fault tolerance in the age of blockchains*. PhD thesis, 2016.

[CL+99] Miguel Castro, Barbara Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.

[CLH+18] Min Chen, Wei Li, Yixue Hao, Yongfeng Qian, and Iztok Humar. Edge cognitive computing based smart healthcare system. *Future Generation Computer Systems*, 86:403–411, 2018.

[DA14] Manab Kumar Das and Samit Ari. Ecg beats classification using mixture of features. *International scholarly research notices*, 2014, 2014.

[EAHL16] Ariel Ekblaw, Asaph Azaria, John D Halamka, and Andrew Lippman. A case study for blockchain in healthcare:"medrec" prototype for electronic health records and medical research data. In *Proceedings of IEEE open & big data conference*, volume 13, page 13, 2016.

[Ecl] Eclipse mosquitto. https://mosquitto.org/. Accessed: 2020-06-22.

[EDST+18] Christian Esposito, Alfredo De Santis, Genny Tortora, Henry Chang, and Kim-Kwang Raymond Choo. Blockchain: A panacea for healthcare cloud-based data security and privacy? *IEEE Cloud Computing*, 5(1):31–37, 2018.

[EIPH18] N El Ioini, C Pahl, and Sven Helmer. A decision framework for blockchain platforms for iot and edge computing. SCITEPRESS, 2018.

[Eur17] European Society of Intensive Care Medicine. Virtual 3d tour of an icu, 2017. [Online; accessed April 16, 2020].

[GJR+15] Tuan Nguyen Gia, Mingzhe Jiang, Amir-Mohammad Rahmani, Tomi Westerlund, Pasi Liljeberg, and Hannu Tenhunen. Fog computing in healthcare internet of things: A case study on ecg feature extraction. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 356–363. IEEE, 2015.

[HTSC08] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. Mqtt-s—a publish/subscribe protocol for wireless sensor networks. In *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE'08)*, pages 791–798. IEEE, 2008.

[IPG+20] Aida Ismailisufi, Tomo Popović, Nenad Gligorić, Sanja Radonjic, and Stevan Šandi. A private blockchain implementation using multichain open source platform. In *2020 24th International Conference on Information Technology (IT)*, pages 1–4. IEEE, 2020.

[Kag] Arrhythmia on ecg classification using cnn. https://www.kaggle.com/gregoiredc/arrhythmia-on-ecg-classification-using-cnn. Accessed: 2020-05-22.

[KFS18] Mohammad Kachuee, Shayan Fazeli, and Majid Sarrafzadeh. Ecg heartbeat classification: A deep transferable representation. In *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 443–444. IEEE, 2018.

[MM01] George B Moody and Roger G Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[MQT] What is mqtt? http://mqtt.org/faq. Accessed: 2020-06-22.

[PBF] Pbft message exchange. http://tokenpost.kr/assets/uploads/20181214678b4a21d41f042cd.png. Accessed: 2021-06-15.

[PKP19] Sunny Pahlajani, Avinash Kshirsagar, and Vinod Pachghare. Survey on private blockchain consensus algorithms. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–6. IEEE, 2019.

[Pyt] paho-mqtt. `https://pypi.org/project/paho-mqtt/`. Accessed: 2020-06-22.

[Rab17] Kefa Rabah. Challenges & opportunities for blockchain powered healthcare systems: A review. *Mara Research Journal of Medicine and Health Sciences*, 1(1):45–52, 2017.

[RGN+18] Amir M Rahmani, Tuan Nguyen Gia, Behailu Negash, Arman Anzanpour, Iman Azimi, Mingzhe Jiang, and Pasi Liljeberg. Exploiting smart e-health gateways at the edge of healthcare internet-of-things: A fog computing approach. *Future Generation Computer Systems*, 78:641–658, 2018.

[RN10] Muhammad Mahbubur Rahman and Afroza Nahar. Modified bully algorithm using election commission. *arXiv preprint arXiv:1010.1812*, 2010.

[Sah18] Sumit Saha. A comprehensive guide to convolutional neural networks — the eli5 way, December 2018. [Online; Accessed: 29-July-2020].

[Sat17] Mahadev Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.

[Sta17] Alexandru Stanciu. Blockchain based distributed control system for edge computing. In *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pages 667–671. IEEE, 2017.

[STÅK18] Per Skarin, William Tärneberg, Karl-Erik Årzen, and Maria Kihl. Towards mission-critical control at the edge and over 5g. In *2018 IEEE International Conference on Edge Computing (EDGE)*, pages 50–57. IEEE, 2018.

[SYB+14] David Schwartz, Noah Youngs, Arthur Britto, et al. The ripple protocol consensus algorithm. *Ripple Labs Inc White Paper*, 5(8), 2014.

[XYYZ19] Fangmin Xu, Huanyu Ye, Fan Yang, and Chenglin Zhao. Software defined mission-critical wireless sensor network: Architecture and edge offloading strategy. *IEEE Access*, 7:10383–10391, 2019.

[Zer] Ømq - the guide. `https://zguide.zeromq.org/`. Accessed: 2020-10-22.

[ZF15] Qi Zhang and Frank HP Fitzek. Mission critical iot communication in 5g. In *Future Access Enablers of Ubiquitous and Intelligent Infrastructures*, pages 35–41. Springer, 2015.

[ZL19] Shijie Zhang and Jong-Hyouk Lee. Analysis of the main consensus protocols of blockchain. *ICT Express*, 2019.

[ZSWL18] Peng Zhang, Douglas C Schmidt, Jules White, and Gunther Lenz. Blockchain technology use cases in healthcare. In *Advances in Computers*, volume 111, pages 1–41. Elsevier, 2018.

[ZXD+18] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, Xiangping Chen, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 14(4):352–375, 2018.