**PRENATAL HEALTH MONITORING SYSTEM USING FOG COMPUTING AND LOAD BALANCING**

PROJECT

***Submitted by***

BL.EN.U4CSE21193 SPOORTHI M

BL.EN.U4CSE21171 RICHA VIVEK SAVANT

BL.EN.U4CSE21176 S NAVIN SUNDER

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

IN

COMPUTER SCIENCE AND ENGINEERING



AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

**BENGALURU 560 035**

MAY - 2024

**Annexure 1**

**PRENATAL HEALTH MONITORING SYSTEM USING FOG COMPUTING AND LOAD BALANCING**

A PROJECT REPORT

***Submitted by***

SPOORTHI M

RICHA VIVEK SAVANT

S NAVIN SUNDER

***in partial fulfillment for the award of the degree***

***of***

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING,

AMRITA VISHWA VIDYAPEETHAM

**BENGALURU 560 035**

MAY 2024

**AMRITA VISHWA VIDYAPEETHAM**

**AMRITA SCHOOL OF COMPUTING, BENGALURU**



**BONAFIDE CERTIFICATE**

This is to certify that the Project Report (19CSE312 – Distributed Systems) entitled “**PRENATAL HEALTH MONITORING SYSTEM USING FOG COMPUTING AND LOAD BALANCING”** submitted by

SPOORTHI M BL.EN.U4CSE21035

RICHA VIVEK SAVANT BL.EN.U4CSE21040

S NAVIN SUNDER BL.EN.U4CSE21057

in partial fulfillment of the requirements for the award of the **Degree Bachelor of Technology** in “**COMPUTER SCIENCE AND ENGINEERING** “is a Bonafide record of the work carried out under my guidance and supervision at Amrita School of Computing, Bengaluru.

SIGNATURE

Ms. Sreebha Bhaskaran

Dept. of CSE, ASE, Bengaluru

This project report was evaluated by us on ……………….

**ACKNOWLEDGEMENTS**

The satisfaction that comes with successfully completing any endeavor would be incomplete without acknowledging the people who made it possible, and whose constant encouragement and direction have served as a source of inspiration throughout the course of this project work.

We offer our sincere pranams at the lotus feet of **“AMMA”,** **MATA AMRITANANDAMAYI DEVI** who showered her blessings upon us throughout the course of this project work.

We owe our gratitude to **Prof. Manoj P.**, Director, Amrita School of Engineering, Bengaluru.

We thank **Dr. E. A. Gopalakrishnan,** Principal and Chairperson-CSE, Amrita School of Computing, Bengaluru for his constant support and inspiration.

It is a great pleasure to express our gratitude and indebtedness to our course faculty **Ms. Sreebha Bhaskaran** ,Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru for her valuable guidance, encouragement, moral support, and affection throughout the project work.

We would like to thank express our gratitude to project panel members for their suggestions, encouragement, and moral support during the process of project work and all faculty members for their academic support. Finally, we are forever grateful to our parents, who have loved, supported, and encouraged us in all our endeavors.

|  |  |
| --- | --- |
| SPOORTHI M | BL.EN.U4CSE21193 |
| RICHA VIVEK SAVANT | BL.EN.U4CSE21171 |
| S NAVIN SUNDER | BL.EN.U4CSE21176 |

**Abstract**

The shift towards continuous care is straining medical professionals' time, burdening healthcare and long-term social care systems. In underdeveloped regions, accessing essential healthcare is hindered by distance or cost. Current monitoring devices lack connectivity, or if connected, pose security risks by storing unprocessed data on third-party clouds. On-premise edge computing at hospitals could address privacy concerns by processing data locally, aggregating information from various sources, and extracting pertinent data for improved healthcare delivery.

Utilizing IoT, e-health devices, and wearable technology enables real-time health tracking, providing valuable insights and alerts. Edge processing ensures secure data exchange between patients and healthcare providers while minimizing costs and bandwidth pressures. It facilitates right-time notifications to practitioners, 360-degree patient dashboards, and secure data storage in the cloud, enhancing resource efficiency, increasing productivity, and reducing costs per patient.

Consistent monitoring during pregnancy is crucial due to the impact of a woman's lifestyle on maternal and fetal health. Pregnancy complications, posing risks like preterm birth and low birth weight, necessitate thorough surveillance. Wearable sensors, m-health technologies, and mobile apps have emerged to monitor fetal and maternal health, tracking parameters such as ECG, heart rate, blood glucose levels, blood pressure, and other biometric data. These devices facilitate real-time data transmission to healthcare providers, establishing a two-way communication channel between doctors and patients to mitigate risks and enhance prenatal care.

Various sensors can collect data as needed, which can then be processed by an edge device and transmitted to the cloud for further analysis and storage. Data preprocessing methods, including filtering to eliminate noise and feature extraction from signals, can be employed. Subsequently, clustering-based approaches can identify abnormal changes in bio-signal data. The Edge technology can facilitate immediate alerts to the healthcare providers regarding any abnormalities. Clinicians can then observe a comprehensive patient dashboard, which displays all the necessary extracted information.

**Table of Contents**

|  |  |
| --- | --- |
| **S.NO** | **CONTENTS** |
| CHAPTER 1 | Introduction |
| CHAPTER 2 | Literature Survey |
| CHAPTER 3 | System Requirements |
| CHAPTER 4 | System Design and Implementation |
| CHAPTER 5 | Testing and Results |
| CHAPTER 6 | Conclusion |

**CHAPTER 1**

1. **Introduction**
   1. Background

The transition to continuous care makes demands on long-term healthcare systems. For instance, in regions faced with underdevelopment, peoples’ access to healthcare is challenged by distance or the cost of travelling to medical facilities. Such challenges can be reduced by Internet of Things (IoT), e-health devices and wearable technology that allows real-time tracking of a person’s health by providing important insights, alarms and alerts. Carrying out data processing near the point where it was generated ensures privacy for sensitive information exchanged between patients and healthcare providers. The low latency features supports real time alerts that are vital in situations such as possible strokes. Moreover, fog computing can cut down costs and help relieve bandwidth pressures by taking care of data close to their origin thereby reducing the volume sent over to health care providers and housed in secure locations.

* 1. Motivation

Throughout pregnancy, a woman's health and the well-being of her fetus are affected by her lifestyle, behavior and physical activity. Continuous monitoring is crucial, especially if potential risks or complications exist.

Pregnancy complications pose significant risks to the health of mother and child and lead to adverse outcomes such as miscarriage, premature birth, stillbirth, and low birth weight. Many wearable sensors, mobile health technologies, mobile applications, and other wireless devices have been developed to monitor fetal and maternal health and physical activity and reduce risks during pregnancy.

1.3 Problem Statement

In prenatal care, continuous monitoring of vital signs such as heart rate, temperature, and blood pressure is critical to the health of pregnant women. However, current healthcare systems often face challenges in providing real-time monitoring and timely intervention, especially in areas with limited medical facilities. Gaps in health care access can lead to significant delays in identifying and addressing underlying health issues, posing risks to mothers and their unborn children. Therefore, there is an urgent need for a prenatal health monitoring system utilizing fog and edge computing that can effectively monitor these vital signs in real time and ensure timely alerts and interventions to improve maternal and fetal health outcomes.

iFogSim is a powerful simulation toolkit designed to model and simulate resource management technologies in Internet of Things (IoT), edge and fog computing environments. It offers many features that make it a valuable tool for researchers and developers in the field. iFogSim provides a graphical representation of the setup, allowing users to clearly visualize the infrastructure and its components. It reliably predicts various parameters that are critical for monitoring system performance. In addition, iFogSim allows defining infrastructure, placing services and optimizing resource allocation, ensuring efficient management of computing resources. The toolkit is based on the Sense-Process-Actuate application model and is ideal for various IoT applications that require real-time data processing and decision-making.

1.4 Objectives

The main objectives of this report are as follows:

* Develop an On-Premise Edge Computing System for Healthcare Data Processing
* To implement the simulation on iFogSim for optimal Load Balancing

**CHAPTER 2**

1. **Literature Survey**

This report is based on a comprehensive review of the existing literature on fog computing solutions in the healthcare industry. Use scientific journals, conference papers, industry reports, and relevant websites to gather necessary information. Case studies and real-world observations are also included to provide a comprehensive understanding of how fog computing is implemented and its impact on healthcare delivery. These sources provide insight into the benefits, challenges, and future prospects of integrating fog computing into healthcare systems, ensuring a comprehensive analysis of its potential to improve patient care and operational efficiency.

2.1 Review Findings

Asghar et al. (2021) show that there is considerable interest in the development of fog-based architectures and load balancing systems (LBS) for health monitoring systems, aiming to reduce latency and improve network efficiency. The study compared LBS to pure cloud approaches and other fog-based approaches and showed that LBS effectively reduced latency and lowered network utilization. The authors highlight the key role of fog computing in improving the performance of health monitoring systems. Their results indicate that cloud computing is not suitable for health monitoring systems requiring real-time responses due to inherent latency. By placing resources at the edge, fog computing reduces latency issues. In addition, efficient data processing requires load balancing among Fog nodes. The proposed LBS not only improves performance, but also highlights the potential of fog computing to significantly improve health monitoring systems by ensuring timely and effective data management and response.

Abeera Ilyas et al. (2022) in their paper “Software architecture for pervasive critical health monitoring system using fog computing” show an argument that suggests that health monitoring would be improved through Internet of Things (IoT) solution which could effectively identify relevant streaming data, but it overlooks latency and quality of service (QoS) concerns. The fog-based architecture improves on latency performance as well as network utilization in health care systems using load balancing. A hybrid architecture is proposed in this study, combining cloud and fog layers showing huge latency decreases and lowered network utilization for IoT based healthcare. In the future, scholars may wish to look at how blockchain technology and machine learning can make health monitoring data more secure.

Kamini Pareek et al. (2021) in their paper “Fog Computing in Healthcare: A Review” the concepts of fog computing in healthcare are discussed and compared end to end. The security challenges and remedies related to data security, privacy, and health monitoring have been addressed in relation to fog enabled facilities. The investigation recommends the use of IoT systems that rely on fog computing for improving health support services as well as latency reduction in healthcare applications. In summary, this conclusion points out the advantages of using fog-based architectures instead of conventional cloud-based platforms like enhanced data security among others. The research supports the application of fog computing towards solving various problems experienced in health care systems.

Ahmad Malekian Borujeni et al. (2017) in their paper “Developing and Evaluating a Real time and Energy Efficient Architecture for An Internet of Health Things” analyzes the patient’s data, IoT devices are used in healthcare industry with processing elements for analyzing signals and updating alerts. For enhancing patient satisfaction, network metrics evaluations compared with Quality of Experience (QoE) from healthcare apps matters most. To improve patients’ care, I suggest a study that involves an IoT based healthcare monitoring system and effective data analysis as well as application placement strategies. Research on better health services by concentrating on patient satisfaction through optimizing QoE while considering network metrics.

Yousef-Awwad Daraghmi et al. (2022) in their paper “Edge–Fog–Cloud Computing Hierarchy for Improving Performance and Security of NB-IoT-Based Health Monitoring Systems” investigated on a clever medical system utilizing NB-IoT technology with the aim of cutting down delay and beefing up authentication protocols. Accordingly, it suggests a hierarchical architecture for remote health monitoring called edge, fog and cloud layers to show the enhanced performance. The conclusion is that this three-layered arrangement helps to lower latency and execution time significantly through better security enabled by efficient authentication protocols. It also emphasizes on Light-Edge protocol as the most efficient way of reducing computation time and transmission delays across all levels of the system.

Gopinath A R et al. (2022) in their paper “Design of Fog Based Remote Health Monitoring System” presented in this study is an architecture for a fog computing based Real Time Internet of Things (IoT) services enabled Remote Healthcare Monitoring (RHM) system. It utilizes local decision making to exploit benefits of fog computing and combines body sensors and cameras for health issue diagnosis. The objective of the system is to improve overall system intelligence, energy consumption, network utilization and reaction time and as such provide patients with safe and credible medical platforms. In summary, the author recommends that future research should explore areas related to fog/cloud-based RHM systems combining wearable sensors with cameras targeting improved health diagnostics and fog computing for real-time IoT services. In protecting data from unauthorized access, it emphasizes on use of encryption techniques and watermarking methods. The RHM system facilitates faster reaction times, less network usage, higher system intelligence, greater energy efficiency in healthcare monitoring.

Aiswarya S et al. (2022) in their paper “Internet of Health Things: A Fog computing Paradigm” proposes an architecture with three tiers that is boosted by fog computing to address the problem of latency in healthcare IoT. It combines fuzzy logic and reinforcement learning to reduce network latency and improve traditional cloud computing techniques. The goal of the project is to make existing healthcare IoT systems better by virtualization as well as machine learning, which cut down on their network usage, computation, and communication latency. In conclusion, this article points out some benefits of a three-tier fog computing architecture for improving health care IoT systems through reducing network latencies. It also highlights several challenges including interoperability, data security, as well as node agility that should be resolved to guarantee the success of fog architecture development. When integrated into healthcare environments, these technologies will go a long way in changing the manner in which clinicians deliver care to patients.

Fatma H. Elgendy et al. (2021) in their paper “Fog-based Remote in-Home Health Monitoring Framework” introduces a framework that uses fog computing to monitor patients. This technology integrates sensors, cameras and data analysis which makes it possible for them to monitor health issues and alert care givers if there is an emergency as well as store encrypted data. The aim is to make the system more efficient and enhance its performance by reducing latency when compared to traditional cloud-based models. Finally, the paper concludes by noting the efficiency of the fog-based approach in remote health monitoring than cloud-based systems with emphasis on improved reaction time, energy consumption, latency and network utilization. Simulation results further confirm that fog-based approach can effectively be used in healthcare surveillance.

Rydhm Beri et al. (2021) in their paper "A novel fog‑computing‑assisted architecture of E‑healthcare system for pregnant women" discuss on a novel fog computing structure that has been designed to monitor pregnant women’s health by incorporating IoT sensors, fog nodes and cloud servers to facilitate efficient data processing. It goes beyond being an accurate system at 98.75% but also provides recommendations on health, immediate alerts, transfers secure data as well as authenticating it. This makes it more effective in improving the results of healthcare of both parties compared to previous studies it gives warning signs if any health matter arises. For this reason, such E-healthcare system highlights on secure data transmission, authentication and immediate alerts on health issues which utilizes fog computing for real time monitoring and alerts besides integrating IOT sensors, fog nodes and cloud servers. Consequently it have proven to improve healthcare outcomes for pregnant women.

B.Priyanka et al. (2021) in their paper "IOT Based Pregnancy Women Health Monitoring System for Prenatal Care

" analyze on the problem of insufficiency of medical information sharing in rural locations, which is particularly noted in prenatal care for mothers. It proposes an IoT system that can use many sensors such as blood pressure, temperature, heart rate and accelerometers to monitor maternal health. This device is designed to offer real-time health data utilizing ARDUINO UNO and IoT technologies thus enhancing prenatal care with a possible reduction to maternal death rates. Through the use of a microcontroller, this device monitors vital signs including temperature, fetal kick count, pulse rate and heart rate by giving alarms on any abnormality which may require immediate attention. The system effectively transmits data through smartphone application using IoT technology which enables remote monitoring that can improve the outcomes of maternity among women.

2.2 Observations

Based on our review of the literature, the papers collectively articulate the effectiveness of fog-based architectures in combination with Load Balancing Schemes (LBS) that help to reduce latency and improve network efficiency. They bring out the importance of IoT devices in health monitoring although they fall short on issues such as latency and Quality of Service (QoS). Furthermore, the paper appreciates fog computing for attending the matter of data security and diagnostics, while NB-IoT technology as well as proposed three-tier architectures aim at reducing further latency and optimizing network usage. These developments highlight the potential of fog computing and IoT integration in improving patient care.

**CHAPTER 3**

**3. System Requirements**

**3.1 HARDWARE:**

* Processor: Intel Core i3 or more.
* RAM: 4GB or more.
* Hard disk: 256 GB or more.

**3.2 SOFTWARE:**

* Operating System: Windows 10, 7, 8.
* IDE: Eclipse IDE
* Java Version: jdk 1.8
* Additional Libraries:
* Cloudsim-3.0.3.jar
* Cloudsim-3.0.3-sources.jar
* Cloudsim-examples-3.0.3.jar
* Cloudsim-examples-3.0.3-sources.jar
* Commons-math3-3.5.jar
* Guava-18.0.jar
* Json-simple-1.1.1.jar

**CHAPTER 4**

1. **System Design and Implementation**
   1. **Installing the Eclipse IDE and Setting up iFogSim**

For setting up the iFogSim environment in Eclipse, there are many steps to follow comprising of downloading the needed files, configuring your Eclipse project and making sure all dependencies are set correctly.

* Step 1: Install Eclipse IDE
* Download & Install Eclipse from the website’s download page.
* Step 2: Download iFogSim
* Visit the iFogSim Github page.
* Either download the repository as ZIP file or clone it using Git.
* Step 3: Set Up Eclipse Project
* Open Eclipse
* Run the executable file of the downloaded zip archive or execute eclipse folder in case you were using Linux or Mac operating system.
* Create a New Java Project (File > New > Java Project).
* Step 4: Importing iFogSim Source Code
* Right click on your recently created project then select import..
* Choose General>File System option then click next..
* Navigate to where you saved (or cloned) ifogsim
* Select src folder and any other required folders such as modules and examples
* Click on ‘Finish’
* Step 5: Resolve Dependencies
* Ensure that all the necessary libraries are included in the build path. In the directory jars, you will have to include the necessary JAR files including cloudsim, commons-math3 and others required by iFogSim.
* Step 6: Running the Simulation
* Create a Run Configuration:
* Go to Run > Run Configurations.
* Create new Java Application configuration.
* Set Project as your iFogSim project.
* Set Main class with one of the examples in org.fog.test.perfeval package (e.g., Health\_topo1).
* Click Apply and Run.
* If everything is set up properly, then simulation will run and you will see output on Eclipse console.

**4.2 Designing Scenario-Based Topologies**

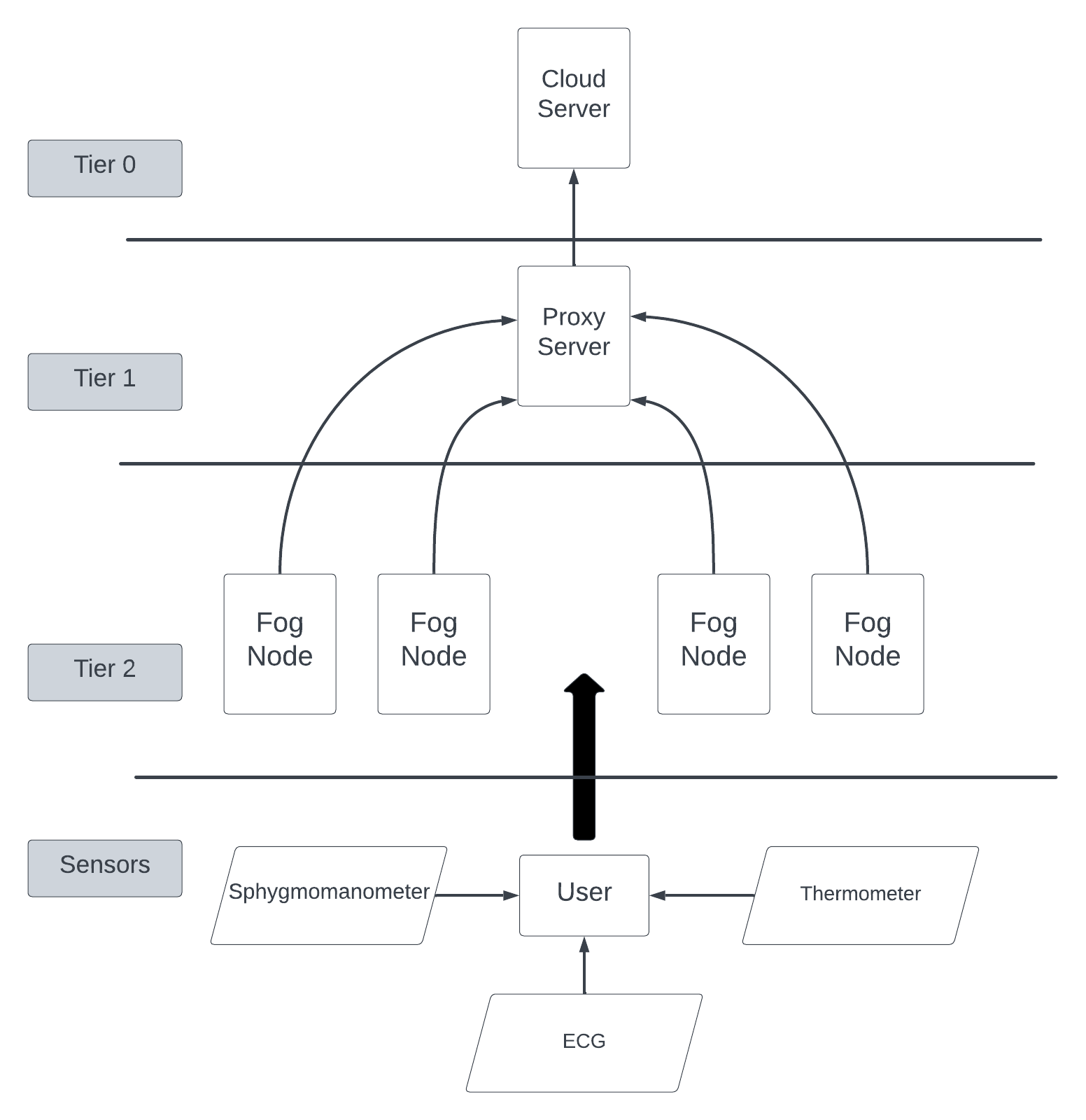
The architecture proposed for the purpose of the study is composed of a hierarchical structure that is divided into four different tiers:

* Tier 0: Cloud Server
* Architecture Position: The highest layer.
* Function: It serves as a central repository and processing hub for data. In spite of its high computational and storage capabilities, it has high amounts of latency due to its geographical separation from end-users and devices.
* Role: The more complicated data analysis, long-term storage and also acts as backup for the data processed at lower levels.
* Tier 1: Proxy Server
* Architecture Position: A cloud server and fog nodes are divided by a middleman layer.
* Function: Decreases delay in the transmission of information from one location to another while at the same time storing some data and performing intermediate processing. It also coordinates communication between the cloud server and the fog nodes.
* Role: To make transfer of data easier, reduce burden on cloud servers and improve response time to requests from the fog nodes.
* Tier 2: Fog Nodes
* Architecture Position: At the edge of the network, below the proxy server and above the sensors.
* Function: Performs localized computation and storage tasks. It reduces latency by bringing data processing closer to where it is generated.
* Role: Supports real-time data processing and analytics that enable immediate responses and decisions. This layer is a must for applications that are time-sensitive.
* Tier 3: Sensors
* Architecture Position: The lowest level having a direct contact with the physical world.
* Function: Gathers raw data from the material world. For prenatal health monitoring, these sensors capture important health parameters from users’ bodies.
* Role: Sends collected information to fog-nodes for initial processing. These include considered sensors for this application:

1. **Sphygmomanometer:** Measures blood pressure.

2. **Thermometer:** Measures body temperature.

3. **ECG Machine:** Measures heart rate and rhythm.



**Fig (4.1)**

Wearable edge sensors that continuously gather data on vital health metrics are incorporated into the design to facilitate prenatal health monitoring. Every tier plays a distinct part in making sure that this data is processed quickly and effectively.

The system guarantees low-latency, real-time monitoring and processing by employing this multi-tiered architecture, which is crucial for providing efficient prenatal healthcare. In order to provide a reliable and effective monitoring system, this design makes use of the advantages of each tier, from instantaneous data processing at the fog nodes to thorough analytics in the cloud.

This architecture is ideal for the crucial job of monitoring fetal health because it guarantees scalability, lower latency, and effective data processing.

4.2.1 Scenario 1:

In order to process healthcare data collected from these three types of sensors in a distributed manner, the system uses fog computing. The data can then be sent for further analysis or storage on the cloud. The core components of the topology include:

* Sensors: Each area/ fog node is equipped with various sensor types to monitor vital health data. This is at the lowest level of the hierarchy. The provided scenario include 3 sensors per area:

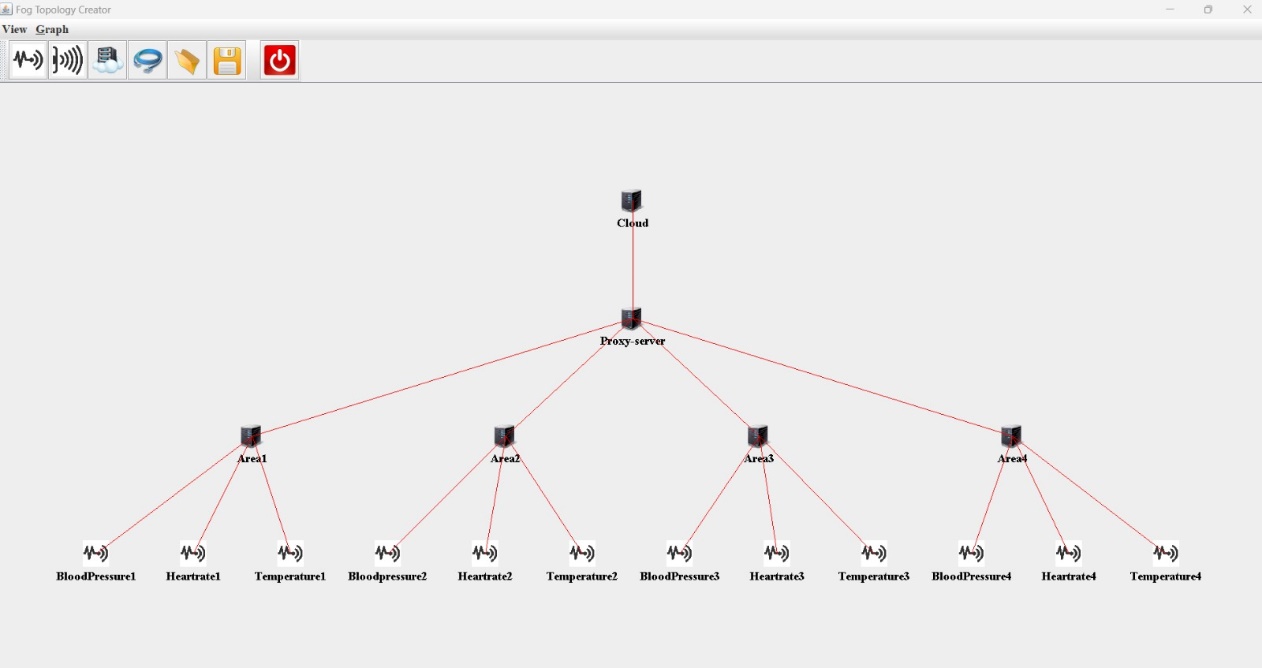
1. Sphygmomanometer: To measure the blood pressure of the mother.
2. Thermometer: To measure the body temperature of the mother.
3. ECG: Electrocardiogram, to measure electrical activity of the heart of the mother.

Every sensor is defined with certain set of pre defined values for particular parameters based on what kind of sensors they belong to (Normal, Deterministic or Uniform)

* Fog Nodes: These devices act as intermediate processing units between the sensors and the cloud server. They are responsible for collecting data from nearby sensors, performing initial processing and analysis, and potentially forwarding the data to other fog devices or the proxy server. The fog devices in this topology have pre-defined processing power (mips) and memory capacity (ram), along with specified upload and download bandwidth (upBw and downBw). In this particular topology, the system is divided into 4 fog nodes, which are specified as “Areas”, (Area1, Area2, Area3, and Area4). The fog nodes/ areas lie at tier 2.

The fog devices are interconnected, forming a fog network. This allows for local data processing and collaboration between fog devices before potentially sending data to the cloud. This local processing can help reduce latency and network traffic, especially for time-sensitive healthcare data.

* Proxy Server: The fog network connects to a central proxy server which is placed at tier 1. This server acts as a gateway between the fog devices and the cloud server. It likely aggregates and manages data flow from the fog network before forwarding it to the cloud.
* Cloud Server: The cloud server marks the utmost hierarchy level and it is rated tier 0. It offers more storage and processing capabilities for health data. Cloud server can be used for more complex tasks such as long-term data analysis, machine learning or data visualization while fog devices perform initial processing.



**Fig (4.2)**

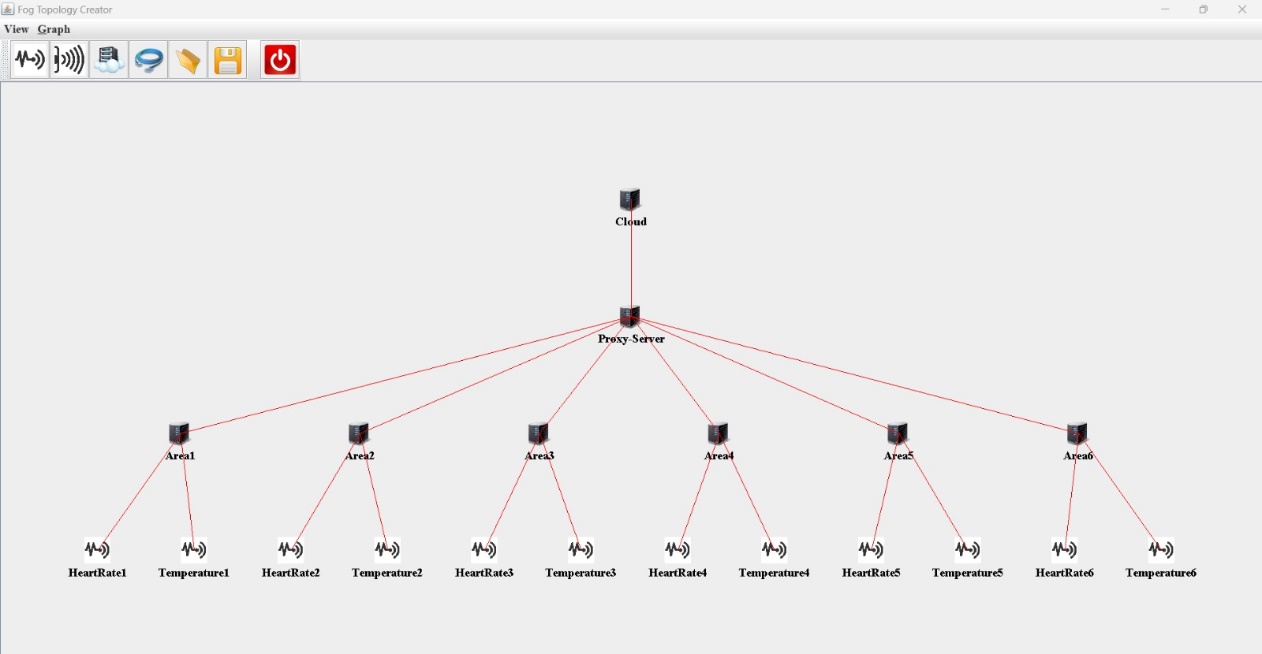
4.2.2 Scenario 2

At tier 0, the Cloud server is placed, which is created as a fog device by defining parameters, such as Uplink bandwidth, downlink bandwidth, RAM, MIPS and rate per MIPS, accordingly.

At tier 1, lies the Proxy server which serves as an interconnecting link between the fog nodes and the cloud. The proxy server has the same parameter fields as that of cloud, but the values differ. The latency of the connection between the proxy server and the cloud is given as 100ms.

The tier 2 consists of fog nodes or areas which in the case of this scenario is 6 fog nodes, the areas again contain the same parameter fields which take in different values for every variable. The latency of the connections between tier 1 and tier 2 is 2ms.

The sensors lie in the lowermost level of the hierarchy. There are 12 sensors taken into consideration for this particular scenario, where in 6 of them are ECGS to measure the heart rate of the mother, and the remaining 6 of them are thermometers. The sensors are assigned specific values for their respective parameters based on the type of the sensor (deterministic, normal or uniform). The thermometer is considered at a normal sensor and the ECG is considered as a deterministic sensor, for which the intervals in which readings are taken is mentioned. The latency of the connections between the sensors and the fog nodes are again defined as 2ms.



**Fig (4.3)**

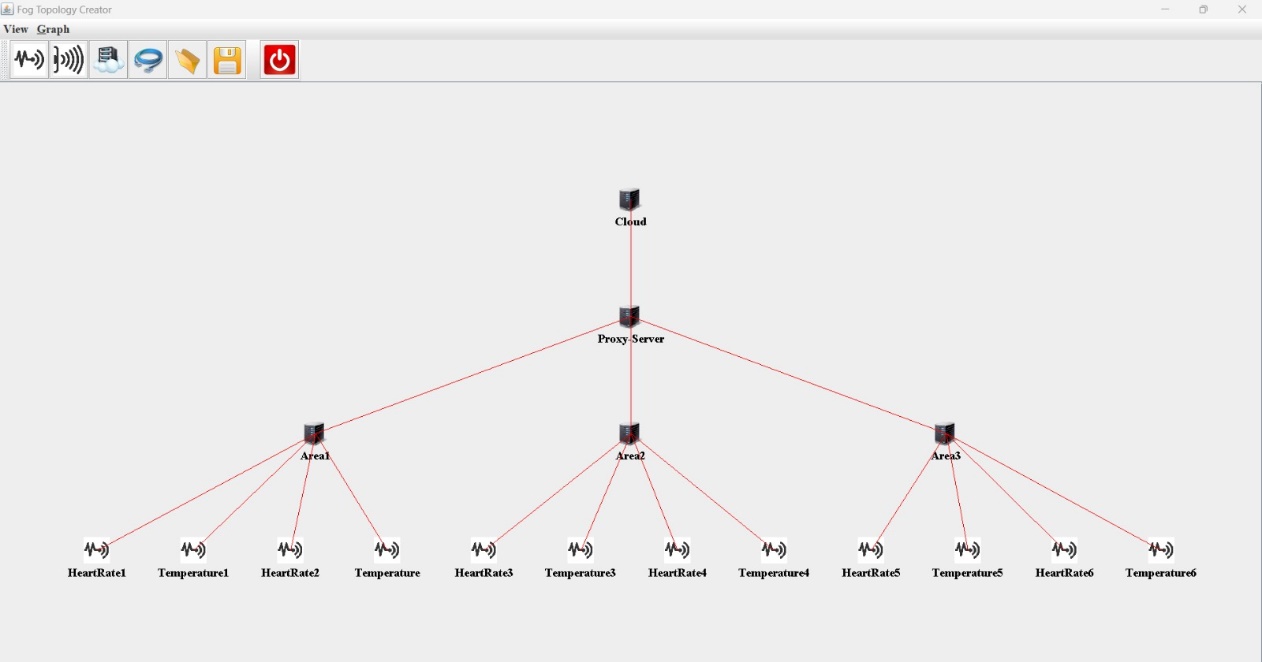
4.2.3 Scenario 3

At tier 0, the Cloud server is placed, which is created as a fog device by defining parameters, such as Uplink bandwidth, downlink bandwidth, RAM, MIPS and rate per MIPS, accordingly.

At tier 1, lies the Proxy server which serves as an interconnecting link between the fog nodes and the cloud. The proxy server has the same parameter fields as that of cloud, but the values differ. The latency of the connection between the proxy server and the cloud is given as 100ms.

The tier 2 consists of fog nodes or areas which in the case of this scenario is 3 fog nodes, the areas again contain the same parameter fields which take in different values for every variable. The latency of the connections between tier 1 and tier 2 is 2ms.

The sensors lie in the lowermost level of the hierarchy. There are 12 sensors taken into consideration for this particular scenario, where in 6 of them are ECGS to measure the heart rate of the mother, and the remaining 6 of them are thermometers. Every fog node is connected to 4 sensors ( 2 ECGs and 2 thermometers). The sensors are assigned specific values for their respective parameters based on the type of the sensor (deterministic, normal or uniform). The thermometer is considered at a normal sensor and the ECG is considered as a deterministic sensor, for which the intervals in which readings are taken is mentioned. The latency of the connections between the sensors and the fog nodes are again defined as 2ms.



**Fig (4.4)**

* 1. **Implementation of the Topologies**

In this setup, fog computing plays a role where it gathers information from different body sensors distributed all over, processes it, and also might carry out an analysis in a centralized manner then sent to the cloud if need be for more processing or storage. The code defines several functions that specify various characteristics of fog devices like:

1. MIPS: Processing power, this is what defines the device's ability to compute.
2. RAM: It stands for the storage available on the device, though not directly specified here, fog devices can have their storage volume designated.
3. Up/Down Bandwidth: These set forth the pace with which data moves back and forth between a device.
4. Level of Hierarchy: The network structure shows how far each device is placed within it.

A FogDevice object is created using the createFogDevice function with these parameters. The Pe list is created internally as part of the function which will represent the processing cores. A PowerHost object is then created within the function that models power consumption based on workload and idle states in addition to architecture, operating system and costs (derived by combining these elements) with others such as Pe list. The function also creates a complete instance for FogDevice which includes all these characteristics.

Network Hierarchy:

The hierarchical network structure has four levels delineated in code:

1. Cloud, which is placed at tier 0.
2. Proxy-server, which is placed at tier 1.
3. Area, which is also referred to as the fog node and represented by "area-X" (with X as the area number) are placed at tier 2.
4. Fog Devices as Sensors: each area has fog devices which represent different body sensors. The sensor fog devices for ECG (Electrocardiogram) data collect heart rate data through "hr-X" devices, blood pressure data is collected by "bp-X" devices and body temperature data is collected by "temp-X" devices. In terms of processing power and storage, these sensor fog devices have lower capabilities than area fog devices. It is probable that they are configured to collect sensor data periodically and transmit it to the area fog device for initial processing or forwarding.

Data flow and application logic

The code defines an application called "health" using the Application class. This program consists of several modules that represent functions in the system:

• sensor module: this module ("ecg", "bp" and "th") receives data directly from the corresponding fog sensor device.

• Processing module: heart rate, blood pressure and temperature modules will perform initial processing or analysis on the received sensor data.

• Decision module: the slot-detector module can analyze the combined data from all sensor modules and make key decisions or trigger actions.

• Actuator module: The "PTZ\_CONTROL" module interacts with the actuator based on the decision of the "locator" module (which is not explicitly defined in this code snippet).

Application logic is defined through the AppEdge object. This side represents the flow of data between different modules in the application. The code defines the direction from the sensor module to the corresponding processing module.

The overall code structure consists of:

1. Library Import: The code imports libraries needed to create fog tools, applications, and iFogSim environment simulations.

2. Global variables: The code defines a sequence such as a flag that indicates the timing of data reading and cloud-based deployment.

3. main function: This function is the entry point for the simulation. It performs the following functions:

* Disable log output for cleaner simulation performance.
* Start the iFogSim environment with user and calendar settings.
* Create a "Health" program.
* Define fog devices using the createFogDevices function. It defines the module deployment logic

Load Balancing in iFogSim:

iFogSim provides functions for load balancing and module placement in the fog computing environment. This feature helps distribute the workload across all cloud devices and optimize resource usage. iFogSim does not have a built-in load balancing module. However, it allows developers to implement load balancing policies in their application logic.

Module placement policy:

iFogSim provides various module placement policies that determine how application modules are mapped to the fog device when the application is deployed.

1. ModulePlacementMapping: This policy uses a static, predefined mapping of application modules to the fog device, set by the user. It does not dynamically balance the load or adjust based on current conditions.

2. ModulePlacementEdgewards: This policy places modules closer to the edge of the network to minimize latency. It prioritizes edge devices over cloud resources, but does not explicitly implement FCFS or Round Robin.

3. ModulePlacementOnlyCloud: As the name suggests, this policy will place all modules in the cloud. This is useful for comparing default values, but does not perform dynamic load balancing.

4. ModulePlacementGreedy: This policy places modules based on resource availability and tries to use the fog node with the most available resources at the time of placement. While not explicitly FCFS or Round Robin, it dynamically responds to resource availability.

By implementing custom load balancing strategies and applying appropriate module deployment policies, developers can optimize application performance and resource utilization in the iFogSim fog computing environment.

The code implemented uses Module Placement Edgewards (MPE) principles for a smart pregnancy healthcare system. This is probably chosen for the following reasons:

1. Real-time data processing: The system works with health data such as ECG, blood pressure and temperature, which are often time-dependent. MPE prefers to place modules closer to sensors (edge ​​devices) to minimize data transmission latency and enable faster analysis for timely decision making.

2. Reduced Network Load: By processing data closer to the source, MPE helps reduce overall network traffic in the fog network. This is essential to ensure reliable data transmission and avoid congestion, especially for critical healthcare data.

3. Privacy Concerns: Processing sensitive health data closer to edge devices may offer some privacy benefits by potentially reducing the amount of data transferred to the cloud or central servers.

**CHAPTER 5**

**Results:**

The three topologies considered for the simulation were implemented on the iFogSim environment by giving various parameter values, as discussed in the description of the example scenarios. The implementation resulted in the output for energy consumption at every node, the energy consumed by each fog device, the total execution time for the simulation of the particular topology. The output values for the different fields were compared for all the three scenarios:

* Scenario 1: This topology has 4 areas that have 3 sensors each, 1 ECG, 1 thermometer and 1 sphygmomanometer.
* Scenario 2: This topology has 6 areas that have 2 sensors each, 1 ECG and 1 thermometer at every fog node
* Scenario 3: This topology has 3 areas that have 4 sensors each, 2 ECGs and 2 thermometers at every fog node.

The three topologies upon simulation were tabulated for their outputs. It was observed that, the topology 1 resulted in the most optimized outputs for the execution times and energy consumption overall. The topology 2 results in high execution time due to more number of fog nodes being used. Although the energy consumption is comparatively less than that of topology 1, the execution time makes it less feasible for a practical implementation. The topology 3 results in huge energy consumption which is comparatively higher than that of topology 1. This is also infeasible for a practical implementation.

High Energy Consumption can have negative impact on the system in:

* Increased operational costs: Fog computing relies on a network of distributed devices. Higher energy consumption at individual nodes translates into higher electricity bills to support the system. This can be a significant cost factor, especially for large-scale installations.
* Reduced sustainability: Excessive energy consumption contributes to a larger carbon footprint. iFogSim aims to use hardware-constrained resources to operate efficiently. High energy consumption goes against the principles of sustainable computing.
* Energy Management Challenges: High energy consumption often translates into heat production at the fog point. This can lead to thermal throttling, where devices reduce processing power to prevent overheating. This can further reduce the performance and execution time of the application.

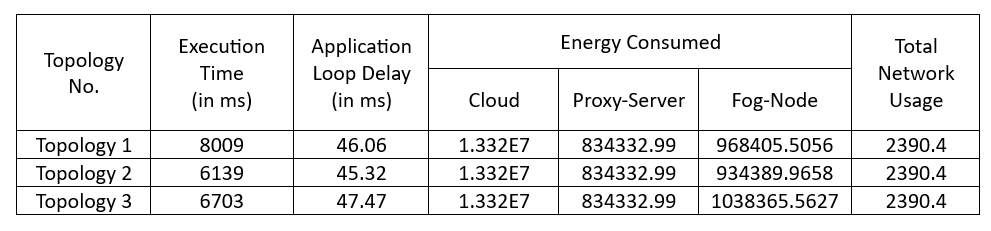
Maximum execution time affects the system negatively in:

* Delay in making decisions: Real-time processing of sensor data is important in applications such as smart healthcare. High execution times at nodes can cause delays in processing and analyzing sensor data. This can affect the system's ability to make appropriate decisions based on the most up-to-date information, potentially disrupting patient care.
* Reduced system throughput: If individual nodes take longer to execute, the overall system throughput suffers. This means that the system can handle fewer requests or issues per unit of time, resulting in potential disruption to users and service delays.
* Increased network latency: High execution times can result in large queues of tasks waiting to be processed on too many nodes. This can cause data to be buffered for longer periods of time, causing higher network latency and affecting system responsiveness.

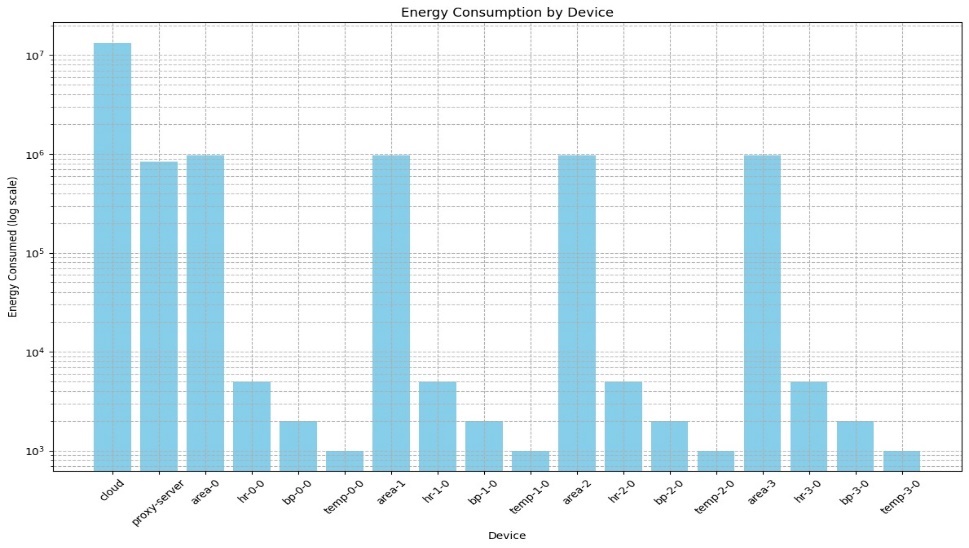
The impact on the system due to high execution time and high energy consumption:

* Degraded performance: High energy consumption and high execution time contribute to a decrease in overall system performance. Users may experience slower response times, longer wait times, and service interruptions.
* Limited scalability: If high energy consumption is a concern, it may limit the ability to expand the fog network by adding more nodes. Likewise, high execution times can prevent systems from efficiently handling increased workloads.
* Decreased user experience: Ultimately, this problem can lead to a poor user experience. Processing delays, poor performance, and potential outages can frustrate users and hinder the effectiveness of fog computing systems.

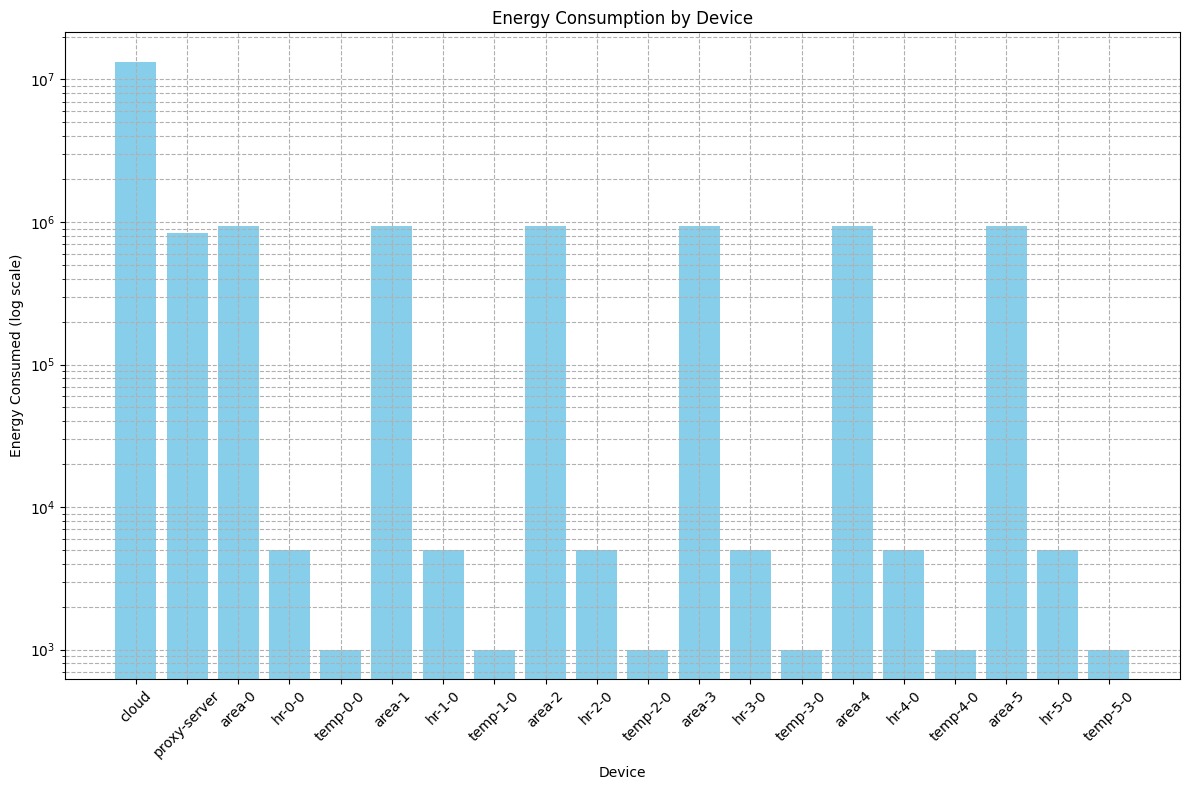
In conclusion, it is important to monitor power consumption and execution time to ensure efficient and consistent operation of the iFogSim system. By optimizing these factors, developers can create cloud-based applications that deliver high performance, scalability, and a positive user experience. Hence, the topology 1 is chosen as the most ideal model for the system implementation.



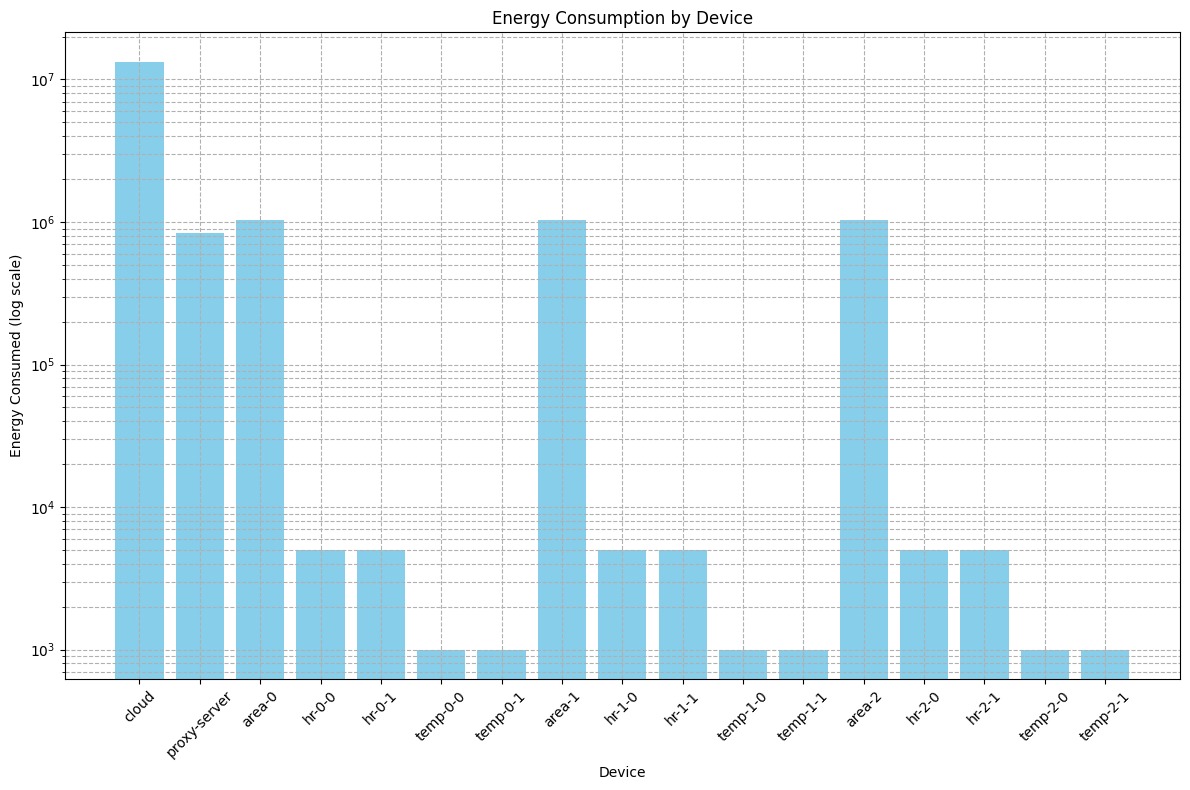
**Fig (4.5)**



**Fig (4.6)**



**Fig (4.7)**



**Fig (4.8)**

Fig 4.6, Fig 4.7, Fig 4.8 represents the values for the energy consumption at every fog device from tier 0 to the sensors for the topologies implemented in Scenario 1, Scenario 2 and Scenario 3 respectively.

From the analysis of the various bar charts as well as the execution times and delays in every topology, it is evident that the topology 1 where three fog nodes are each connected to each if the three types of the sensors (1 ECG, 1 thermometer, 1 sphygmomanometer) gives the most ideal values for energy consumption as well as an ideal execution time. Both the parameters considered for evaluation is less in topology 1 and hence is chosen as the most ideal topology design.

**CHAPTER 6**

**Conclusion And Future Scope:**

This study sought to identify the best design for energy use and execution time by examining three fog computing topologies through iFogSim. The best choice was scenario 1 which had three fog nodes connected to 3 different sensors, each of a different type. Compared to Scenarios 2 and 3, Scenario 1 uses the least amount of energy in terms of optimality on the fog devices. Hence, this leads to lower costs of operation as well as smaller carbon footprints which are consistent with iFogSim’s sustainability ideologies. Of all the tested topologies, scenario one takes a shorter time in its implementation than any other, enabling real-time processing of sensor data indispensable for smart healthcare applications. Also, faster execution enhances system throughput and reduces network latency. Scenario 2, though it consumes lesser energy than scenario one nevertheless it has more fogs nodes therefore its productivity is much lower due to increased decision making process time lapse, low throughput levels and network delay. Scenario 3, even though it has almost similar times of execution like scenario one but still its power consumption is very big when compared with that of scenario one. As a result, it is not suitable for deployment in the real world, as this will lead to high operating costs and sustainability issues. So basically, on that note, we can say that in Scenario 1, energy efficiency and execution time are perfectly balanced thereby making it the best fog computing topology for this particular application. On other hand, this design enhances performance of a system through high scalability as well as minimizing the amount of time spent in processing data.