

## APPLIED RESEARCH

# Design and Simulation of an Edge Compute Architecture for IoT-Based Clinical Decision Support System


**RACHURI HARISH KUMAR**  **AND BHARGHAVA RAJARAM**

Department of Electrical and Computer Engineering, École Centrale School of Engineering, Mahindra University, Hyderabad 500043, India

Corresponding author: Rachuri Harish Kumar (harish20pee001@mahindrauniversity.edu.in)

**ABSTRACT** Clinical Decision Support Systems (CDSS) have revolutionized healthcare by leveraging modern technologies such as internet of things (IoT), artificial intelligence (AI), predictive analysis, nanomedicine, and virtual & augmented reality. IoT-based CDSS is of interest in particular. In a hospital scenario, a patient's vital signs like heart rate, blood pressure, respiration rate, ECG, EEG etc. are monitored with the use of embedded sensor devices, also called smart medical devices. These devices collect real-time data which is relayed to a compute device where several algorithms are employed to perform computations on said data to arrive at a prognosis e.g. real-time onset of hypotension can be detected by running predictive algorithms on real-time blood pressure data. The computation in IoT-based CDSS is done predominantly on the cloud, wherein the real-time data collected is relayed to a centralized cloud server. However, latency is a major drawback in a cloud-based monitoring system. Increased latency is of greater concern in healthcare applications as the decision-making process is time-sensitive. Edge computing can potentially overcome this drawback, wherein computation is done on edge-network devices rather than the cloud. While edge computing for IoT-based CDSS has been explored in literature, there are gaps in their implementations. A majority of literature dealing with edge computing for IoT-based healthcare only demonstrates a single application and does not address the varying data acquisition rates for different vital signs. Each prognosis or diagnosis requires different subsets of vital signs, and the underlying algorithm uses different sizes of data e.g. detecting arrhythmia requires processing of ECG data which is a time series data, and detecting cardiovascular disease requires blood pressure, cholesterol and certain habits of the patient which are mostly single points of data. This paper explores the use of edge computing in CDSS, quantifies its performance with respect to number of devices, sense time interval or intertransmission rate, and the size of data, and proposes a unified IoT edge gateway architecture to combine multiple patterns of data and computation algorithms to achieve reduced latency and network utilization. Simulation results show that edge computing reduces the latency of decision by approximately 87 times, and the network utilization by 1.5 times. The results show the efficacy of edge computing for implementing IoT-based CDSS and also demonstrate scalability with regard to the number of devices and the size and intertransmission rate of data.

**INDEX TERMS** Edge computing, smart healthcare, vital sign monitoring, Internet of Things, clinical decision support system.

The associate editor coordinating the review of this manuscript and approving it for publication was Tawfik Al-Hadhrani .

## I. INTRODUCTION

Clinical Decision Support Systems (CDSS) are used in healthcare to provide stakeholders of the healthcare system including doctors, nursing staff, and patients with information

necessary to make efficient decisions [1]. A traditional CDSS is a software system which aids in clinical-decision-making wherein the diagnostic results and characteristics of patients are compared with computerized databases, and patient-specific assessments or recommendations are then presented to the doctor for a decision. The aforementioned characteristics of a patient are primarily their vital signs including temperature, heart rate, respiration rate, and blood pressure [2] and other physical characteristics, including height, weight, gender, ethnicity, prior health conditions etc. [3]. These characteristics can also include the results of diagnostic tests performed on a patient. Healthcare professionals use this data to make informed decisions regarding the patient's wellness.

Although CDSS has been used by healthcare professionals for a few decades, recent technologies such as Internet of Things (IoT), blockchains, artificial intelligence, high-performance computing etc. have increased the efficacy of the system with respect to the range of decisions, accuracy, and the time taken to make the decisions [4]. IoT, in particular, provides a methodology to acquire the required data from patients using sensors, store this data, and make decisions using the acquired and historical data using machine learning algorithms, and relay the decisions to the end user, which includes the healthcare professional, insurance companies, the government etc.

An IoT-based CDSS is typically implemented through the integration of smart medical devices and cloud-based systems, enabling informed decisions to be made based on real-time patient data. With the increasing use of smart medical devices, the growth of IoT-based CDSS is predicted to accelerate rapidly [5]. However, this growth also means that there will be an increased burden on cloud computing resources to compute the huge amount of real-time data generated by these devices.

The quantity of data generated by a patient in intensive care units (ICUs) can vary depending on the number of smart medical devices connected to the patient, which can range from 1 gigabyte to terabytes of data per day. Analyzing this data in real-time is necessary. Although sending the total amount of data to the cloud can be an effective strategy for analysis, it carries several risk factors such as high latency, power consumption, security and bandwidth requirement.

A potential concern with transmitting all the generated data to the cloud for analysis is the network and computation latency which can delay the decision-making process. This could be problematic in time-sensitive scenarios, where delays in data communication and analysis could negatively impact patient outcomes. Additionally, the use of cloud resources to analyze the data can require significant bandwidth, and a stable internet connection. Additionally, server downtime caused by maintenance and cybersecurity breaches can increase latency further. Although various features have been introduced by cloud service providers, there are still several risks associated with storing data on external or third-party servers. This makes the centralized

cloud-based computing model unsuitable for time-sensitive applications.

As shown in Figure 1, the fundamental architecture of IoT-based healthcare comprises of three layers: the sensor layer at the bottom, the gateway layer in the middle, and the cloud layer on top. The function of these layers is as follows:

- The **sensor layer** is responsible for gathering raw data from the patient's body through various sensors.
- The **gateway layer** acts as the communication interface between the sensor layer and the cloud layer and it translates non-IP packets to IP packets to transmit the data to the cloud over traditional IP protocols layer for analysis.
- Finally, the **cloud layer** is responsible for analysing the data and making decisions.

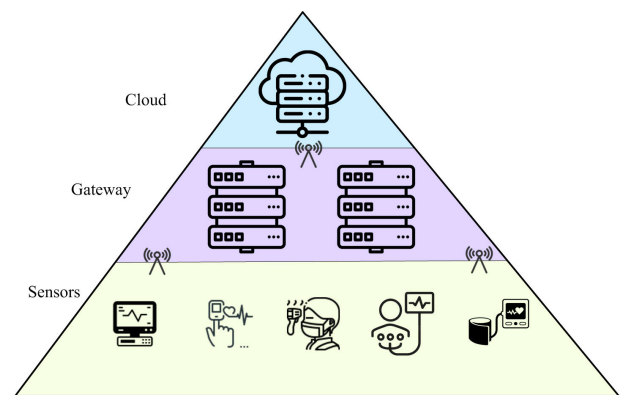


FIGURE 1. Gateway enabled IoT architecture.

However, because the cloud layer is often located in a geographically remote area from the patients, it can lead to increased latency. To reduce latency, cloud computing power must be brought closer to the sensor layer.

In this paper, we propose an edge-compute-based IoT architecture for a CDSS that incorporates edge computing at the gateway layer. Performing the computation in the gateway layer helps reduce latency. Although computing power can also be integrated at the sensor level, doing so may result in increased size, cost, power consumption, and maintenance requirements for the sensor. The proposed architecture was simulated using iFogSim [6] to obtain latency and network usage by varying the number of devices, size of the data, and the data acquisition rate also referred to **sense time interval or inter-transmission rate** in subsequent sections. We quantify sense time interval as the amount of time that elapses between two successive data transmissions from the same sensor node. Simulation results indicate a significant improvement in network usage and latency as compared to the cloud-based clinical decision support system. we have simulated four different healthcare applications, such as

- Hypotension
- Cardiovascular disease
- Arrhythmia
- Combination of above three applications

**TABLE 1. Improvement of performance of the proposed edge compute gateway over cloud-based architecture.**

Application	Patient to Alert Latency	Network Usage
Hypotension	96.6X	1.27X
Cardiovascular	95.04X	1.37X
Arrhythmia	86.86X	4.33X
Combined	72.54X	3.38X

Table 1 shows the comparison between edge computing with cloud computing for sensor to actuator (patients to alert system) latency and network usage.

As the table shows, sensor-to-actuator (patient-to-alert) latency has improved drastically for all applications. Network usage also shows significant improvement over cloud-based solutions. The proposed gateway architecture, which supports multiple applications with varying data patterns and compute algorithms, also shows significant improvement. This validates both the usage of edge computing and the proposed framework for healthcare applications where latency is critical.

The rest of the paper is organized in the following manner: Section II presents recent research on intelligent healthcare monitoring systems, followed by the basic fundamentals of edge computing and its associated literature. Section III proposes the edge compute gateway-based architecture, and Section IV covers the simulation setup. Simulation results are presented in Section V, followed by the conclusion in Section VI.

## II. RELATED WORK

### A. CLINICAL DECISION SUPPORT SYSTEMS (CDSS)

This section discusses the literature related to the critical health care monitoring system. Researchers have proposed architectures and solutions for various parts of CDSS, including methods for acquiring the vital signs of patients, algorithms to analyze the data, studies to correlate vital sign variations with physical ailments, and overall system architectures to implement CDSS.

With the advancement of Artificial Intelligence (AI) [7], cellular communication (5G and beyond) [8], [9], Internet of Things (IoT) [10], [11], Cyber-Physical Systems (CPS) technology [12] and various new computing paradigms [13], [14], integrated healthcare systems are transforming into smart healthcare environments for better treatment with more accuracy. There have been several efforts in recent years to solve the challenges faced in the healthcare sector by using the aforementioned technologies. These include work in healthcare epidemiology [15], [16], Electronic Health Records (EHRs) [17], [18], [19], [20], drug management [21], healthcare monitoring, real-time resource tracking [22], community healthcare [23], and clinical decision support system (CDSS) [24].

There has been a lot of effort in the literature to define the parameters that are useful in CDSS. Dias et al. [25] considered five traditional vital signs that play a major

role in monitoring patients, such as heart rate, blood pressure, respiratory rate, blood oxygen saturation, and body temperature.

Three more vital signs are considered by Ahrens et al. [26] as important to immediately measure the patient condition, i.e. capnography (to evaluate ventilation), pulse oximetry (to evaluate arterial oxygenation) and stroke volume (to evaluate heart function)

Elliott and Coventry et al. [27] described another three important vital signs that should also be considered as part of routine patient assessment-pain, level of consciousness, and urine output.

On the data acquisition front, Mamady Kebe et al. [28] reviewed various methods to track cardio-pulmonary signal detection and further discussed the feasibility of using continuous-wave radar for detecting heart rate variability and respiration rate.

Liu et al. [29] analyzed the condition of patients by considering the major vital signs during sleep time. The authors categorized the methods used to collect the vital signs into four groups:

- dedicated sensor-based
- smartphone and wearable sensor-based
- touch-free sensor-based
- RF signal-based

With regards to the decision-making algorithms used in CDSS, Sutton et al. [1] classified CDSS into two types:

- knowledge-based CDSS, which uses if-then-else rules based on expert knowledge
- AI-based CDSS, which is known as non-knowledge-based

Bashir et al. [30] proposed a knowledge-based CDSS framework to detect cardiovascular disease. Montgomery et al. [31] investigated the effect of a computer-based clinical decision support system and a risk chart on absolute cardiovascular risk, blood pressure, and prescribing of cardiovascular drugs in hypertensive patients. Indeed, to diagnose acute and chronic diseases, vital signs play a crucial role in analysing the patient's condition, as reported by Kaieski et al. [32].

Chester et al. [33] analysed age-related changes in vital signs (blood pressure, pulse, respiratory rate, and temperature) based on changes at the molecular and systemic levels with respect to the organ systems.

Peiffer et al. [34] concluded that the effectiveness of CDSS depends on the accuracy of vital sign data and also the decision-making algorithm.

Finally, on the architecture front, Prajapati et al. [35] proposed an IoT-based healthcare system that can help with fast communication, identify emergencies, and initiate communication with healthcare staff to initiate proactive and quick treatment. Abdelgawad et al. [36] proposed the customized healthcare IoT architecture to collect the sensor data and transmit it to the central cloud for further processing and analysis. A decision was taken based on the analyzed data. Salem et al. [37] defined IoT as a communication

algorithm that enables the exchange of vital data to provide remote data visualization and enable real-time vital data treatment.

Another aspect to consider in smart healthcare systems is data privacy and security. Gia et al. [38] posited that data management in healthcare is one of the most sensitive issues, as user data contains important and private information that needs to be analyzed and processed in a manner that is both efficient and secure.

It is worth noting that all of the IoT-based CDSS implementations consider a cloud-based solution wherein data acquired from patients is communicated to a centralized cloud server where the analysis is done and the resultant decisions are communicated downstream to healthcare providers. A centralized cloud server-based implementation puts more burden on the network and thus results in increased latency in decision-making, which might have adverse effects on the patient in question. In this paper, we focus on IoT architectures for CDSS based on edge computing to reduce the latency of the decision-making process. The next section discusses the concept of edge computing and its applicability to smart healthcare systems.

## B. EDGE COMPUTING FOR IoT

There are several works in literature which argue for the use of edge computing in general and specifically for IoT applications. This section discusses some of the papers which evaluate the applicability of edge computing.

Edge computing is a distributed architecture that connects a range of smart embedded devices indirectly to each other at the end of the network to analyze and store the data to enhance the quality of service [39]. It offers a significant advantage by minimizing the need for frequent access to the cloud for data uploading. As a result, the bandwidth required to communicate with the cloud is reduced, leading to an improvement in response rates. Various studies have demonstrated that edge computing leads to enhanced performance, lower network usage, and reduced latency compared to the cloud in various domains like industries, unmanned aerial vehicles and manufacturing industries [40], [41].

Perez et al. [42] posited that edge computing is a well-suited computing paradigm to accomplish the enormous demand for bulk connections and low response time applications. It achieves this by carrying out specific computation and processing tasks on the edge gateway instead of relying solely on the cloud server.

Edge computing also offers a more flexible platform and a more effective method of data computation by utilising the limited network capacity [43]. Mohammed Laroui et al. [44] concluded that edge computing can provide significantly more data computation power and storage at the edge of the network when compared to the cloud computing paradigm.

Shi et al. [45] explained edge computing as a distributed computing paradigm that brings computation and data storage closer to the location where they are needed. The paper also discussed the advantages of edge computing,

including improved efficiency through local data processing and reduced network congestion, latency, and bandwidth requirements. Additionally, the authors also provided case studies in various domains, demonstrating how edge computing can improve efficiency, reduce costs, and provide better services to end-users. However, healthcare was not one of the applications discussed.

The reference architecture of edge computing for Industrial IoT was proposed by Qiu et al. [46]. The proposed architecture consists of a cloud layer and an edge layer, and the edge layer can also be subdivided into Near-Edge, Mid-Edge and Far-Edge. Frameworks enable efficient and scalable deployment of IIoT systems by leveraging the advantages of edge computing.

Sharma et al. [47] proposed an edge-computing architecture named SoftEdgeNet, which is based on software-defined networking (SDN) principles. The aim of this architecture is to enhance the energy efficiency of network systems by delegating computation and storage tasks from conventional cloud servers to edge devices. The use of SDN techniques in SoftEdgeNet enables dynamic resource allocation and management, which mitigates network congestion, latency, and bandwidth requirements.

Hamdan et al. [48] presented a survey that focuses on edge computing architectures (ECAs) for IoT and how they address various challenges that arise in IoT systems, such as security, scalability, and management. Additionally, the authors mapped ECAs to two existing IoT layered models. By doing so, the authors were able to identify the capabilities, features, and gaps of each architecture in a structured manner.

As discussed above, edge computing offers a low-latency alternative to cloud computing for IoT applications. However, this has not been evaluated for healthcare applications, which are time-sensitive. Also, the existing evaluations of edge computing do not discuss scalability in terms of the data acquisition rate. This paper evaluates edge computing for healthcare applications with varying numbers of devices, data sizes, and sense time interval.

## C. IoT-BASED CDSS WITH EDGE COMPUTING

Nguyen et al. [49] proposed a fully decentralized healthcare architecture for distributed Electronic Medical Records (EMRs) sharing among federated hospitals using emerging technologies blockchain and Mobile Edge Computing (MEC) to enhance the system security and data retrieval rate using decentralized EMRs storage built on an interplanetary file system platform, which is a MEC server. This work focuses on blockchains for decentralization and sharing of medical records, not on making clinical decisions.

Singh et al. [50] introduced an innovative Secure Framework known as SEoT (Secure Edge of Things) for real-time monitoring and emergency services. The authors propose a clustering algorithm for anomaly detection, followed by securing the data using Attribute Based Encryption (ABE) techniques. Singh et al. focus more on the security aspect and not on supporting multiple applications. The same algorithm



is applied to all data. In this paper, however, we propose a configurable framework to support multiple applications and also validate the framework for performance and scalability.

Prabhu et al. [51], presented an Edge-IoT ecosystem Testbed using EdgeX Foundry. EdgeX Foundry is used to leverage edge computing to design a telehealth framework. However, their proposed architecture also implements and demonstrates a single application, i.e., blood pressure monitoring and forecasting.

Li et al. [52] proposed a secured framework for software-defined network-based edge computing in IoT-enabled healthcare systems. This work focuses on providing a framework where edge compute devices collaborate to perform computations on the acquired data. The framework focuses on the SDN framework itself and the load balancing between edge compute devices for a constant data rate, and not on application-based computation or data requirements.

In [53], Graphics Processing Units (GPU) are used as the edge device to train and test the Recurrent Neural Network (RNN) for activity prediction using a wearable sensor-based system, and it compares with conventional approaches on a publicly available standard dataset. This work demonstrates the efficacy of using a GPU as an edge compute device.

Abdell et al. [54] propose an MEC-based architecture for smart healthcare systems to address the challenges of delivering scalable healthcare services while reducing costs. The work presents and demonstrates the improvement provided by edge computing for EEG-EOG signal processing using deep neural networks, wherein encoded features alone are communicated to the cloud server where the final decision-making is done. In our proposal, however, the decision-making happens at the edge device itself to achieve better latency.

Dong et al. [55] propose an edge computing-based healthcare system for the Internet of Medical Things (IoMT) to address the challenges of wireless channel deficiency and limited computation resources. This work focuses more on the communication aspects and bandwidth allocation and not on the computation aspects.

Haoyu Wang et al. [56] proposed a priority-based task queuing method at the edge that enables emergency tasks to be processed earlier. This is a pre-edge layer that only dispatches data and tasks to the edge or cloud based on priority.

#### D. INFERENCES FROM LITERATURE

Existing literature primarily showcases the efficacy of edge computing in healthcare, often focusing on a singular application or emphasizing other aspects such as communication and security. In contrast, our paper introduces a framework for an edge compute gateway designed to support multiple data patterns and compute algorithms.

During the development of this architecture, we encountered a challenge arising from the lack of standardized data types in Internet of Things (IoT) applications. To systematically address this issue, we introduced distinct data

categories, encompassing single point, Temporal, Spatial, and Spatiotemporal data. This comprehensive classification takes into account both the intrinsic characteristics of the data and its processing dynamics at the edge. By integrating these categories, our architecture not only accommodates the diverse nature of IoT data patterns but also establishes a robust foundation for efficient processing and analysis tailored to the specific context of IoT applications. This categorization approach enhances the adaptability and scalability of the architecture, ensuring its relevance across a wide spectrum of IoT scenarios.

### III. PROPOSED SYSTEM ARCHITECTURE

In this section, an edge gateway-based architecture for CDSS is discussed and compared with the traditional cloud-based architecture.

#### A. CLOUD-BASED IoT ARCHITECTURE

As mentioned in the previous sections and shown in Figure 2a, an IoT-based architecture for CDSS consists of three layers, i.e., sensors, gateway, and cloud layer.

- The lowermost layer of the architecture comprises numerous smart medical devices that monitor various parameters of patients. This layer is responsible for generating a packet for each patient containing the patient's id and vital signs data and sending it to subsequent layers for analysis using personal area network (PAN) protocols like Bluetooth, ZIGBEE, LoRAWAN, etc. These smart medical devices are called **sensors** in subsequent discussions.

This layer also consists of devices which can act on the information resulting from the analysis of the data collected by the sensors. These include actuators at the patient side, e.g., regulating IV medication, display systems at nursing stations, or other alert mechanisms. These end devices are called **actuators** in subsequent sections.

- The intermediate layer of the architecture comprises a gateway that communicates with the sensors and actuators and also with the cloud. The function of a gateway is to relay a) data collected by the sensor devices to the cloud and b) decision data from the cloud to the actuator devices.

Gateways are necessary in IoT systems as most of the sensor and actuator devices are non-IP devices, i.e., they cannot communicate directly over the internet. These devices communicate with the gateway over the above-mentioned PAN protocols. The gateways communicate with the cloud using IP-based protocols like Ethernet, cellular communication, WiFi, etc. [57].

- The topmost layer of the architecture is the cloud layer, which is mostly located in a remote geographic location. In traditional IoT architecture, data collected from the sensor devices is analyzed in the cloud using appropriate algorithms, and the resultant decision is sent back to the actuator devices through the gateway.

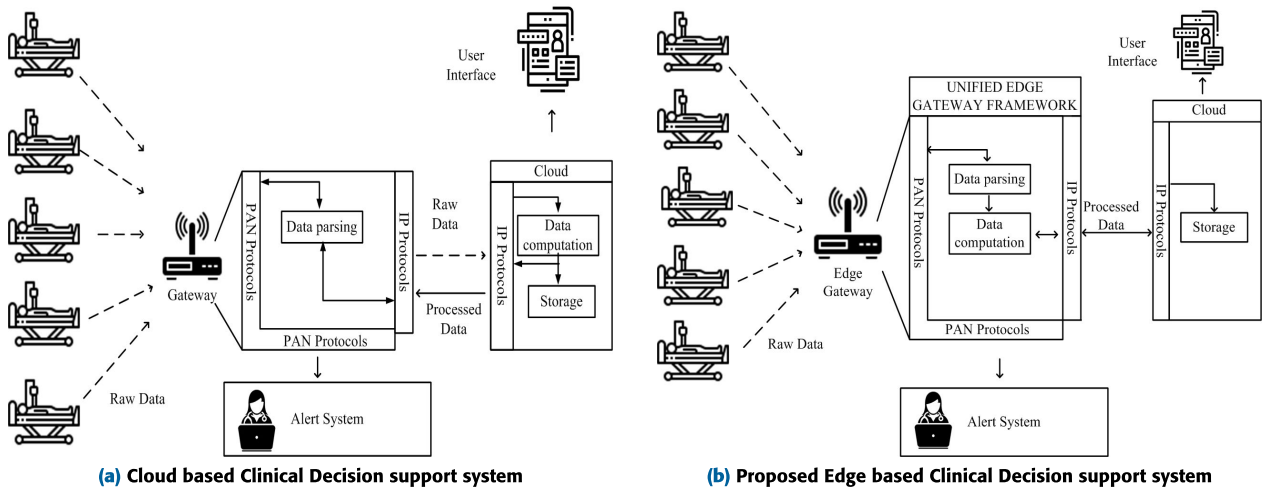


FIGURE 2. Clinical decision support system.

**B. PROPOSED EDGE-GATEWAY-BASED ARCHITECTURE**

The above cloud-based architecture offers the ability to store the information for a longer period, higher computation capability, as well as visualization. However, constant access to the cloud for uploading and downloading information not only increases the latency and network usage but also consumes a lot of power, raising more security concerns.

By introducing the edge gateway as the middle layer, the computations required for finding the abnormalities of a patient are performed at the edge of the network. Latency is reduced because frequent access to the cloud is not required.

Figure 2b shows the proposed edge-gateway-based CDSS architecture for one ICU with a scalable number of patients. Each ICU will have its own edge gateway connected to the centralised cloud server. Network utilisation and latency in this scenario will remain constant for each edge gateway, but they will rise while updating and retrieving data from a centralised cloud server. We assume that computation at the edge will not be spatially spread out, i.e., computation performed on the data acquired from one patient does not depend on data acquired from another patient.

*Security Considerations:* Typical IoT systems employ lightweight encryption protocols to provide security [58]. Apart from encryption, authentication of the devices is also key to fulfilling security requirements. By manually provisioning nodes, the issue of authentication is solved. The nodes and their IDs are sent to the gateway as update commands to the configuration table.

In the case of encryption, the implementation needs additional information on the encryption keys, which can be stored in the configuration table along with the rest of the metadata. Encryption, however, can increase the size of data packets [59]. Our architecture exhibits superior performance in terms of latency, network utilization, packet sizes, number of nodes, and sense time interval. Thus, separate simulations for encrypted data was not performed.

Number of Applications	Application ID1	Application ID2	----- Application IDn	Patient ID	Patient Data
------------------------	-----------------	-----------------	-----------------------	------------	--------------

FIGURE 3. Data packet format.

The following is a discussion of the proposed architecture’s components and their interaction:

1) SENSOR LAYER

This layer is the same as the sensor layer in the cloud-based architecture. Sensor devices attached to the patient first acquire the necessary bio-signals. The acquired signals are conditioned for the removal of noise, selecting frequency regimes, and performing amplification and analog-to-digital conversion as required.

The difference in the edge-based architecture is that the same data point can be used for several diagnoses using different algorithms. For e.g. the measured systolic and diastolic pressures are used to predict both cardio-vascular disease and hypotension, using different inference algorithms. For this purpose, the packets transmitted by the sensor devices to the edge gateway should include the *patient id* and *application id*. Since multiple applications can be supported, the packet data first specifies the number of applications that this data item is to be used for, followed by the list of application ids. The resulting packet data format is shown in Figure 3.

2) EDGE GATEWAY LAYER

The edge gateway layer is the middle layer between the medical devices and the cloud, which is responsible for collecting the raw data from the medical device to further detect the abnormalities in the patient and uploading it to the cloud. Each patient has a specific ID, which also includes the bed assigned. The edge compute gateway updates the patient status to the electronic display, which is placed in the nursing station of the ICU as well as to the cloud for long-term

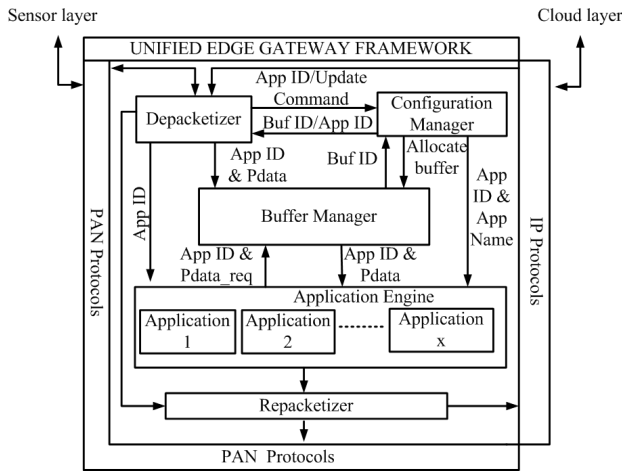


FIGURE 4. Unified edge gateway framework.

storage. The edge gateway performs necessary computations on the received data, transmits the diagnosis to the appropriate healthcare workers, and also transmits the *filtered/processed* data to the cloud.

The architecture of the proposed IoT edge gateway is shown in Figure 4. The data packets are received over PAN protocols from sensor devices, from which the patient ID and set of application IDs are extracted, and the corresponding application engines (algorithms) are instantiated to work on the vital sign data. The processed data, which is the diagnosis, is sent back to the sensor layer and the cloud layer for storage and future analysis.

It is to be noted that the sub-components of the architecture are structured as separate processes to leverage parallelism in the process. Each of the components in the data path are pipelined to increase the throughput of the gateway. Each component of the architecture, except for the PAN protocols and IP protocols, is elaborated with Algorithm in the following sections.

*a: DEPACKETIZER*

The depacketizer is a packet parser processes packets both from the sensor layer and the cloud layer. If the packet is an IP packet ( $(C_{pc})$  or  $(D_{pc})$ ), the depacketizer checks if the packet is an update command ( $(C_{pc})$ ) for the configuration manager or data ( $(D_{pc})$ ) to be sent downstream to the sensor layer. Downstream packets are redirected to the sensor layer after extracting data and repacking it for the corresponding PAN protocol. It is assumed that the packet will contain the identity of the node in the sensory layer that should receive this packet. checks for errors in transmission as per the PAN protocol used.

The depacketizer will extract the sensor data, patient id and the list of application IDs, from the data packet marked as sensor data -  $(S_d)$  It will also retrieve the buffer ID from the configuration manager to facilitate data storage in the allocated buffer space, with the assistance of the buffer manager. The application ID will initiate the application from an idle state to a running state.

**Algorithm 1** Depackitizer

```

Output: Depackitized packet
Input : sensor data packets ( $S_d$ ); command packets from cloud ( $C_{pc}$ ); data packets from cloud ( $D_{pc}$ )
while packet received
do
  if Packet is ( $S_d$ )
  then
    Get BuffID from the configure manager
    Send AppID to application engine
    send data and Buffid to buffer manager
  end
  else if Packet recieved is ( $C_{pc}$ )
  then
    Send AppId/Command to configuration manager
  end
  else if Packet recieved is ( $D_{pc}$ )
  then
    Extract data from ( $D_{pc}$ )
    send extracted data to repackitizer
  end
  else
    Wait for packet to receive
  end
end
  
```

*b: BUFFER MANAGER*

The Buffer manager maintains the mapping of each patient id to the buffer space allocated to that patient id. This is used by the packet parser to store data for the application engines to use accordingly. The buffer manager also allocates the buffer space when the configuration data is updated.

*c: CONFIGURATION MANAGER*

Configuration data maintains a table which provides details on the function names for each application algorithm, the buffer space required for that algorithm, and the identity (address) of the node in the sensor layer that should receive the processed data. The table also mentions whether all of the received data is to be stored in the cloud or only the processed data. The gateway is reconfigured on receiving an update command from the cloud to include a new computation algorithm. Buffer space is allocated for each patient connected to the edge gateway as per the requirements of the computation algorithm used. The configuration manager also creates tasks for each application engine as defined in the configuration table.

*d: APPLICATION ENGINES*

The application engines are the list of tasks created by the configuration manager to operate on the received data. Since the processing done is patient specific, we classify the type

**Algorithm 2** Buffer Manager

---

```

Input : App ID & Pdata,
AppID & pdata_req,
Buff_req.
Output: Buff_ID,
App ID & Pdata.

while input
do
  if Buff_req
  then
    foreach patient ID do
      Allocate buffer space according to new
      requirements
      Update mapping of patient ID to allocated
      buffer space
    end
  end
  if App ID & Pdata
  then
    Extract patient ID from received data
    Store received data in the allocated buffer
    space
  else
    Retrieve allocated data for the patient ID
  end
end
end

```

---

**Algorithm 3** Configuration Manager

---

```

Data: Configuration table with function addresses,
buffer space requirements, node identity, and
data storage preferences

Input : Update command from the cloud
Output: Reconfiguration and task creation
if ( $C_{pc}$ ) includes a new computation algorithm
then
  Add new algorithm details to the configuration
  table
  Allocate buffer space for each patient connected
  to the edge gateway based on new algorithm
  requirements
  Create tasks for each application engine based on
  the configuration table
end
else
  send the Buffer ID to depacketizer
end

```

---

of data received as a) **single point data**, and b) **time series data**. For example, determination of hypotension is done with a single data point, whereas cardiac arrhythmia is detected by processing ECG which is a time series data. The time series

data is first stored in buffers via the packet parser, which in turn uses the buffer manager. The buffer address is passed to the application engine for further processing.

**Algorithm 4** Application Engine

---

```

Data: Received data

Input : Data received from the packet parser
Output: Processing of data

if App ID is single application
then
  Perform task specific to application(e.g.,
  determination of hypotension)
end
else if multiple application
then
  Store data in buffers via the packet parser
  Pass buffer address to the application engine
  Perform task specific to time series data
  processing (e.g., processing ECG for cardiac
  arrhythmia detection as well as Hypotension)
end

```

---

e: *REPACKETIZER*

The repacketizer has two functions: to receive “depacketized” downstream packets from the cloud layer to be sent to the sensor layer using PAN protocols, and to send the processed data to the cloud layer or back to the sensor layer. The PAN protocol used depends on the patient id in the downstream packet or the processed data. This is maintained as a separate mapping table in the gateway in order to support multiple PAN protocols.

## 3) CLOUD LAYER

The prominent role of a centralised cloud in the proposed architecture is to maintain the database of patients and their various medical features for the long term and retrieve data when it is needed by the application. The edge gateway is connected to the cloud server through a proxy server to increase privacy. The edge gateway uploads the sensor information to the cloud after a specific period of time, and if the edge gateway needs some patient information, then the data is recovered from the cloud.

**IV. SIMULATION SETUP****A. SIMULATORS**

In order to evaluate the proposed edge gateway framework for implementing CDSS with respect to performance and scalability, we propose the use of a simulator as opposed to a practical implementation owing to increased infrastructure cost, and difficulties in evaluating scalability. This section describes the simulators used in the literature to evaluate an IoT system that utilizes edge computing.



**Algorithm 5** Repackitizer

---

**Data:** Received packets, Mapping table of patient IDs to PAN protocols

**Input :** Depacketized packets, Processed data

**Output:** Packets sent to sensor or cloud layer

```

while packet
do
  if Depacketized packets
  then
    foreach Depacketized packets
    do
      Extract patient ID
      Determine PAN protocol based on patient ID
      Send packet to sensor layer
    end
  end
  if processed data to cloud layer
  then
    Send processed data to the cloud layer;
  end
  else if processed data to sensor layer
  then
    Extract patient ID (integer)
    Determine PAN protocol based on patient ID
    Send processed data to sensor layer
  end
end
end

```

---

Ashouri et al. [60] discussed various simulators used for simulating the various computing paradigms. One of the simulators for edge and fog computing is iFogSim, which is based on the CloudSim simulator. The performance evaluation of scheduling algorithms is generally performed using simulation tools, including professional simulators for edge computing such as iFogSim [6].

iFogSim is one of the Java-based Free and open-source Source Software tools that supports the evaluation of various computing paradigms environments, such as cloud edge and fog computing paradigms [6]. Several researchers used the iFogSim simulator to evaluate their proposed work for fog and edge-based applications [61]. Baccarelli et al. [62] created a simulation model using the iFogSim simulator as a proof of concept to determine energy-delay performance.

Awaisi et al. [63] simulated an application of fog-based car parking using the iFogsSim Simulator and compared major parameters like network usage and latency.

In this study, we explore latency and network utilization across a variety of scenarios by utilizing the iFogSim simulator as our tool of choice.

To track the various vital signs of the patient, various sensors are considered. The vital signs are eventually transmitted to the computation node. The computation node

analyzes the vital signs to identify the abnormality and displays them on the electronic screen placed in the nursing station, which is connected to the gateway. A proxy server is a virtual node between the gateways and the cloud server.

In our simulated environment, we emulated an intensive care unit, complete with patients and their vital signs as integral components. We focus on simulating various ICU scenarios. In each of these scenarios, we initially deployed 12 sensor nodes in the ICU. Each sensor node is responsible for individual patient vital sign data collection within the ICU. To comprehensively analyze patient conditions, diverse medical parameters were essential, leading to the integration of multiple medical devices into each sensor node. The ICU was equipped with a single gateway, and the proxy server acted as the intermediary linking our gateways to the cloud infrastructure within the environment. This entire setup was configured in accordance with the guidelines mentioned by Harshit Gupta et al. [6]. To assess the computational load on the system, we increased the number of sensor nodes and varying time gap between successive instances of data sensing, commonly referred to as the “sense time interval” from 500 millisecond(ms) to 1 ms, aiming to understand its impact on network usage and latency.

Moreover, simulations were carried out for five different physical topology arrangements, labeled as *Config-1* through *Config-5*. These setups correspond to systems with 12, 24, 36, 48, and 60 sensor nodes, respectively. In each case, a gateway is incorporated into the physical topology for every set of 12 sensor nodes.

The physical topology designed for *Config-2* using iFogSim, as depicted in Figure 5, follows an IoT gateway architecture. This visual representation showcases two separate ICU setups, with each ICU connected to its dedicated gateway. Each gateway is equipped with 12 sensors, an electronic display, and a connection to a centralized cloud through a proxy server. To enhance the complexity of the simulation, we have introduced additional configurations up to *Config-5*.

In the Edge Compute Gateway-based scenario, we’ve embedded machine learning models directly within the gateway to identify anomalies. The gateway takes on the role of the primary computation node in this configuration. It subsequently relays alerts to both the electronic display placed in the nursing station and the cloud, facilitated by a proxy server. Conversely, in the cloud-based scenario, the cloud assumes the role of the central computation node, where machine learning algorithms are deployed for data processing. The processed data is then transmitted to the electronic display via a proxy server and the gateway.

With an increasing number of patients, additional gateways are deployed, and the responsibility of detecting patient abnormalities shifts to the computing nodes, which can be cloud or gateway. However, as the number of sensor nodes connected to a particular gateway increases, there is a corresponding rise in network usage and latency for that specific edge gateway.

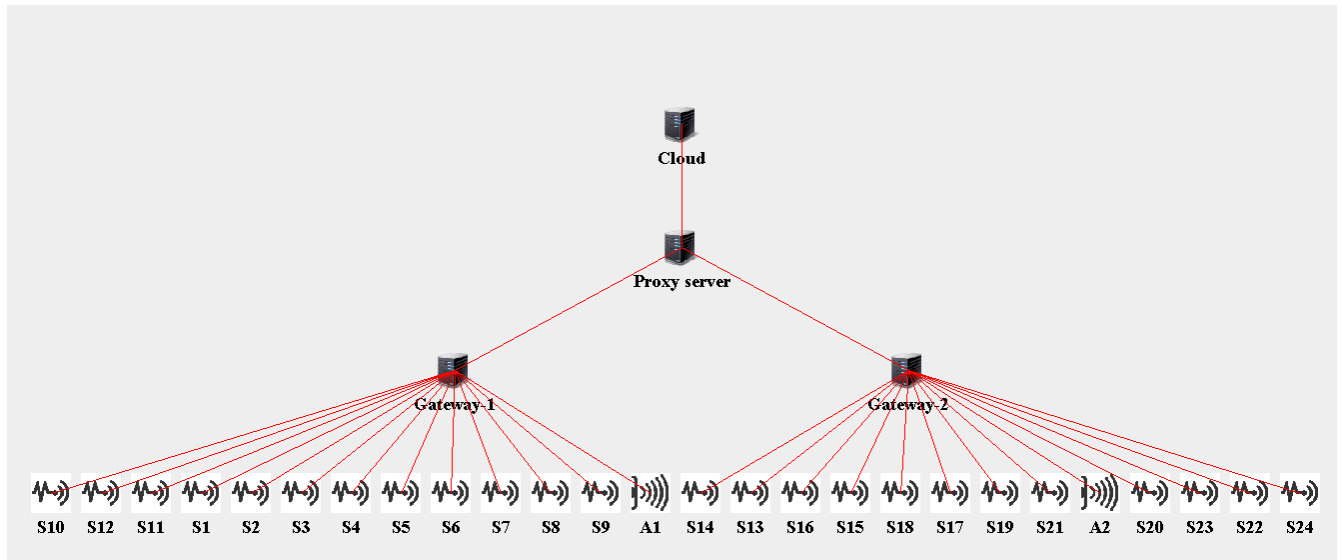


FIGURE 5. iFogSim topology of 24 sensor nodes connected to 2 gateways connected to the cloud server via the proxy server.

TABLE 2. Configurations of edge gateway, proxy server and cloud.

Parameter	Cloud	Proxy	Edge Gateway
CPU length (MIPS)	3680	1800	800
RAM (MB)	2048	1024	512
Uplink bandwidth (MB)	500	500	500
Downlink bandwidth (MB)	500	500	500
Level	0	1	2

The configuration details for the cloud, proxy server, and edge gateway in both the edge compute gateway-based scenario and the cloud-based scenario can be found in Table 2. These setup parameters encompass the CPU capacity in terms of million instructions per second (MIPS), Random Access Memory (RAM) in megabytes, uplink and downlink bandwidth in megabytes, and the architectural level.

In Figure 6, we illustrate the data flow for various applications, including Hypotension, Cardiovascular, and Arrhythmia, designed to detect abnormalities in patient vital signs within our simulation environment. Each application model consists of key components, such as sensors, detector modules, actuators, and user interfaces. It's important to note that the inference algorithms employed in the detector modules differ based on the specific application. In the following sections, we provide detailed explanations for each application module.

### B. APPLICATION MODULE

In iFogSim, an application is depicted as a directed acyclic graph (DAG), with its nodes being represented by modules and its links symbolizing data dependencies. In Figure 6 sensor, detector, actuator, and user-interface modules are shown as vertices, while the arrows connecting these modules to the sensor and actuator represent the edges.

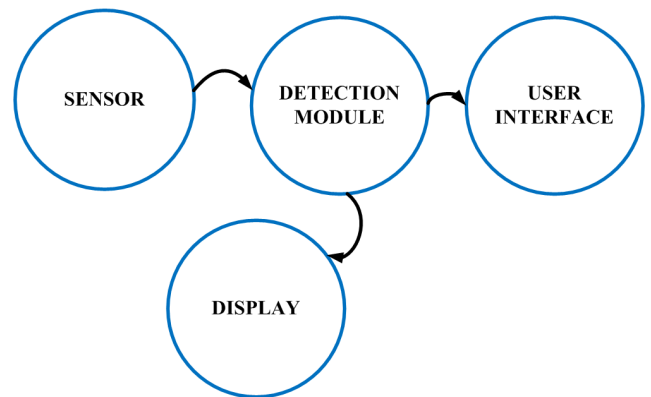


FIGURE 6. Dataflow for individual application.

In iFogSim, we utilize the AppModule class to represent application modules. Within the system, there are inherent data dependencies between these modules, as visualized in Figure 6, and these relationships are effectively modeled in iFogSim through the AppEdge class.

The AppLoop class is instrumental in emulating the control loops that pertain to specific applications. The Sensor module is responsible for supplying vital sign data to the program, while the Display module presents the application's results to the user. The following delineates the roles played by the various components:

#### 1) SENSOR

This class represents a physical component in iFogSim. It functions as the sensor/end node within the iFogSim simulator and is responsible for emulating the behavior of hardware devices such as Electrocardiograms, Blood pressure monitors, or Temperature sensors. It serves as the starting point where tuples are generated to simulate a variety of

tasks or jobs. These tuples are then distributed throughout the network for processing.

Tuples, which are the basic units of communication between different entities, inherit their characteristics from the Cloudlet class of CloudSim. Each tuple is defined by its type, source, and destination modules. The class's attributes include processing requirements, measured in Million Instructions (MI), and the size of data encapsulated within the tuple. Additionally, the class incorporates patient-specific information as parameters, and the tuple data size in bytes is detailed in Table 3. It also features a reference attribute to identify the gateway device to which the sensor is connected, as well as information about the connection's latency. Furthermore, this class defines the output characteristics of the sensor and the distribution of tuple inter-transmission or inter-arrival time, which determines the rate at which tuples arrive at the gateway.

**TABLE 3. Tuple size with respect to application.**

Application	Size of data in bytes
Hypotension	85
Cardiovascular	125
Arrhythmia	1550
Combined	1650

## 2) ACTUATOR

This class functions as an end-node simulator, depicting various hardware devices that become active upon processing the tuples generated by the sensor class. These devices include components like storage systems, motors, furnaces, and others. Notably, in this configuration, the Actuator role is fulfilled by the display.

Furthermore, the display is placed within the Nursing Station to enable real-time monitoring of patients' vital signs, ensuring swift data transmission and display with minimal latency.

## 3) USER INTERFACE

The User Interface is a module hosted in the cloud, allowing for the storage and display of information from any location and at any time.

## 4) DETECTION MODULE

### a: HYPOTENSION

The Hypotension detector module is responsible for detecting abnormalities in vital signals received from the sensor using a machine learning algorithm. This module informs the actuator about Mean Arterial Pressure (MAP) and detection of hypotension conditions to be updated on display in the nurse station.

Hypotension is a critical medical condition characterized by the systemic blood pressure dropping below the established threshold [64]. Blood pressure refers to the force exerted by the heart as it pumps blood against the walls of the arteries. It becomes problematic when this pumping force

is insufficient to adequately supply oxygen-rich blood to the body's vital organs. Consequently, the patient's symptoms begin to impact their quality of life.

Blood pressure is determined by two physiological factors: systolic blood pressure (SBP) and diastolic blood pressure (DBP). Systolic blood pressure changes of less than 90 mm Hg or a mean arterial pressure of less than 65 mm Hg are considered indicative of low blood pressure. Furthermore, a drop in diastolic blood pressure to below 40 mm Hg can also be significant.

In Hypotension conditions, hypotensive shock is a possible and life-threatening condition. Sharma et al. [64] defined blood pressure as the product of cardiac output and total peripheral vascular resistance.

The mean arterial pressure(MAP) is the average blood pressure over the course of one cardiac cycle [65]. It is calculated as equation 1 & 2:

$$MAP = DP + \frac{1}{3}(SP - DP) \tag{1}$$

OR

$$MAP = DP + \frac{1}{3}(PP) \tag{2}$$

where DBP is the diastolic blood pressure, SBP is the systolic blood pressure, and PP is the pulse pressure. This method is often more conducive to measuring MAP in most clinical settings as it offers a quick means of calculation if the blood pressure is known as discussed by Kumari et al. [66]. Annotated dataset incorporating attributes as specified in Table 6. The decision making algorithm to be used in the final application was decided on comparing different classifiers. Figure 7 shows the accuracy of various classifiers for hypotension. As seen from the figure, SVM provides the highest accuracy, and thus was chosen in the final implementation.

**TABLE 4. Clinical attributes to identify the Hypotension in patients.**

S.No	Attribute Name	Type
1	MAP	Integer
2	Gender	Binary
3	Age(years)	Integer
4	Systolic blood pressure	Integer
5	Diastolic blood pressure	Integer

### b: CARDIOVASCULAR DISEASE

The critical application in the healthcare sector focuses on the detection of cardiovascular disease, which involves considering factors such as age, gender, height, systolic and diastolic blood pressure, cholesterol levels, glucose levels, smoking habits, alcohol consumption, and physical activity. To detect cardiovascular disease, patients are required to connect an embedded device and provide the necessary information.

For this study, we used an annotated dataset on cardiovascular disease obtained from Kaggle [67]. The dataset

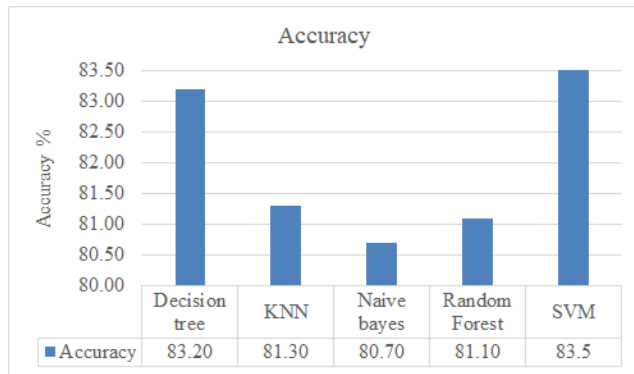


FIGURE 7. Accuracy of various classifiers for hypotension.

comprises eleven properties, and Table 5 illustrates its content. The analysis specifically focuses on individuals aged between 29 and 64, and their weight and height measurements are also recorded, as discussed by Dwivedi et al. [68].

We observed that the decision tree algorithm delivered exhibited more accurate predictions compared to Random Forest and KNN algorithms as discussed by Princy et al. [69]. Figure 8 shows the accuracy of various classifiers for cardiovascular disease. Decision trees was chosen as the classifier for cardiovascular disease prediction in the final implementation of the gateway.

TABLE 5. Clinical attributes to identify cardiovascular disease in patients.

S.No	Attribute Name	Type
1	Age	Integer
2	Height	Integer
3	Weight	Float
4	Gender	Binary
5	Systolic blood pressure	Integer
6	Diastolic blood pressure	Integer
7	Cholesterol	Integer
8	Glucose	Integer
9	Smoking	Binary
10	Alcohol Intake	Binary
11	Physical activity	Binary

c: ARRHYTHMIA

Arrhythmia is characterized by irregular heart rhythms resulting from disturbances in the heart’s electrical signaling. Identifying these irregular rhythms involves a comprehensive approach that encompasses medical evaluations, diagnostic techniques, and monitoring. The primary diagnostic tool for detecting arrhythmia is the electrocardiogram (ECG or EKG), a non-invasive and painless procedure typically conducted in clinical or hospital settings. The ECG records the heart’s electrical activity, allowing the identification of irregular rhythms by capturing and displaying the subtle electrical impulses generated by the heart as a graphical trace on paper.

To facilitate the identification of arrhythmias, the arrhythmia detector module processes the ECG signal. It conducts a series of operations to extract key statistical features, including Mean, Median, standard Deviation, and Skewness.

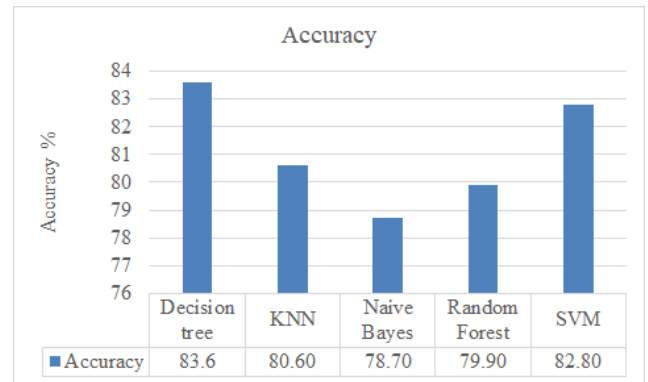


FIGURE 8. Accuracy of various classifiers for cardiovascular disease.

These statistical features are then input into a machine learning algorithm to identify any abnormalities in the ECG signal, as done by Vijaya et al. [70].

I) MEAN

The mean or average is a statistical measure that represents the central tendency of a set of values. It is calculated by dividing the sum of these values by the total number of elements in the set, as shown in equation 3.

$$x_{\mu} = \frac{1}{k} \sum_{i=1}^k x_i \tag{3}$$

II) MEDIAN

The median, which is expressed in equation 4, is the middle value of the given set of values. In other words, the median acts as the dividing point between the upper and lower halves of a given data sample.

$$x = \frac{x_{n+1}}{2} \tag{4}$$

III) STANDARD DEVIATION

The standard deviation, which is expressed in equation 5, serves as a metric for gauging how widely a set of values are distributed around the mean.

$$s = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \tag{5}$$

IV) VARIANCE

Variance, represented in equation 6, serves as a measure of how much the set of values deviate from the mean.

$$s^2 = \frac{\sum (x_i - x_{\mu})^2}{n - 1} \tag{6}$$

V) SKEWNESS

Skewness, represented in equation 7, is a measure used to assess the asymmetry in the distribution of a set of values. It is calculated by taking the sum of the values in the data distribution and dividing it by the total number of values in



that distribution.

$$x_{ske} = \frac{\sum_i^k (x_i - x_\mu)^3}{(n - 1)x_{std}^3} \tag{7}$$

Accuracy of various classifiers for arrhythmia is shown in Figure 9. Similarly to hypotension, SVM was the classifier chosen for arrhythmia.

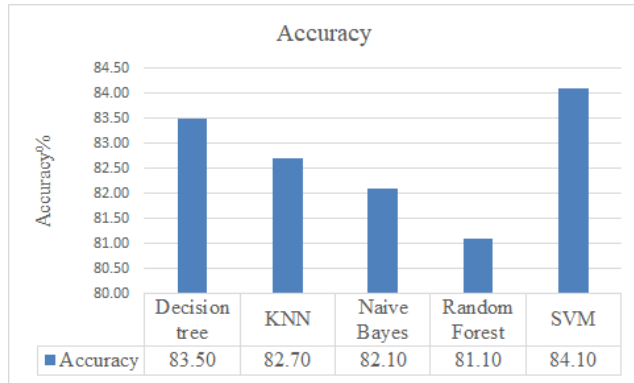


FIGURE 9. Accuracy of various classifiers for arrhythmia.

TABLE 6. Clinical attributes to identify the hypotension in patients.

S.No	Attribute Name	Type
1	MAP	Integer
2	Gender	Binary
3	Age(years)	Integer
4	Systolic blood pressure	Integer
5	Diastolic blood pressure	Integer

### C. COMBINED APPLICATION

In the context of combined application, the computation node is linked to various vital sensors, and these sensors are integrated with five distinct modules, namely:

- 1) Depacktezier Module
- 2) Hypotension Module
- 3) Cardiovascular Module
- 4) Arrhythmia Module
- 5) Repacketizer

The Depacktezier module gathers data from all sensors, as mentioned in the previous section, depacketizes the data, and forwards it to the appropriate modules. For instance, data from the ECG sensor is directed to the arrhythmia module. Within the Arrhythmia module, the ECG sensor data is processed and analyzed to detect abnormalities. The module subsequently generates a response, which corresponds to the actions to be taken based on the ECG sensor values. This response is then conveyed to the user interface for display through the Repacketizer Module.

Figure 10 represents the visual representation of the data flow in this CDSS system application model.

### V. RESULTS AND DISCUSSION

In this section, we delve into the results obtained from comparing the latency and network usage of the proposed

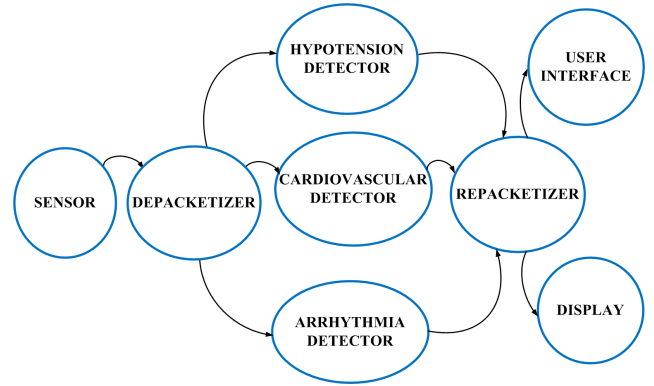


FIGURE 10. Dataflow for combined application.

edge-based architecture with a cloud-based architecture. We evaluate both paradigms in two distinct scenarios:

- 1) By increasing the number of devices.
- 2) By varying sense time interval.

The simulation results for both scenarios demonstrate that the edge-based architecture effectively reduces latency, minimizes network utilization, and enhances overall system performance when compared to the cloud-based architecture.

### A. ANALYSIS OF LATENCY

For applications requiring real-time performance, the reduction of latency is of utmost importance. Edge computing provides a significant advantage by minimizing data transmission to the cloud and conducting data processing at the network’s edge, resulting in swift responses to client devices and consequently a decrease in latency. In the context of an ICU, the vital signs of patients are transmitted to the edge gateway for processing. Each edge gateway is dedicated to a specific ward, ensuring that it possesses sufficient processing power to handle data from that particular facility and rapidly display patient information on electronic display.

To calculate latency, equation 8 is employed, as referenced from [6].

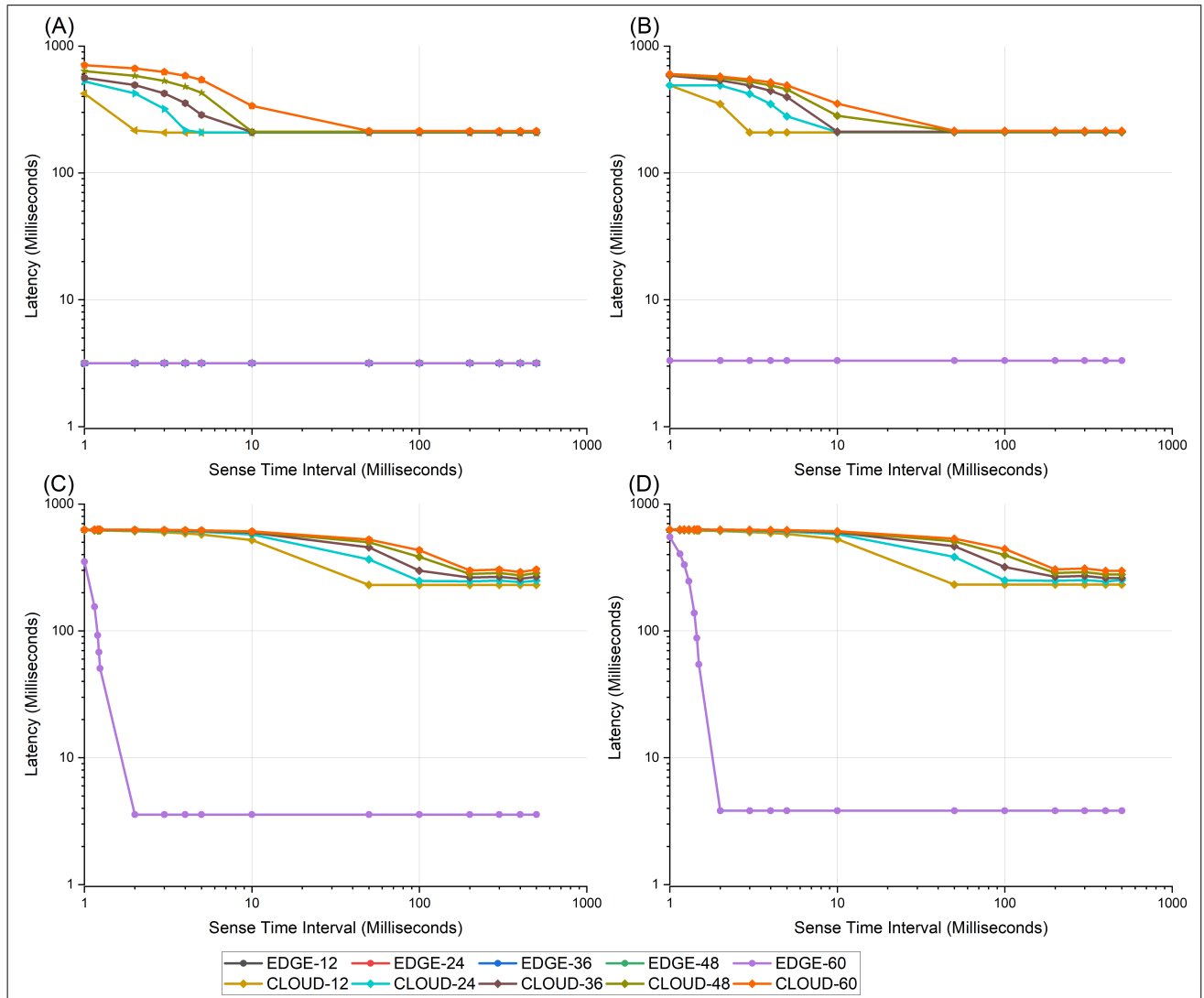
$$\text{Overall Latency} = \alpha + \mu + \Phi \tag{8}$$

$$\mu = \mathbf{F}(\psi, N) \tag{9}$$

where,

- $\alpha$  symbolizes Tuple CPU Execution Delay, indicating the time taken for the processing node (edge/cloud) to process a tuple.
- $\mu$  represents Transmission Latency, signifying the delay incurred during data transmission.
- $\Phi$  stands for Actuator Time, delineating the duration required for an actuator to respond to a command.
- $\psi$  denotes the Sense Time Interval, representing the interval between sensing operations.
- $N$  quantifies the Number of nodes per edge gateway.
- $\mathbf{F}$  pertains to an Empirical Function, implying a function derived from observed data or practical experience.

The Figures 11 and 12 display latency graphs for the communication between sensors and actuators, as well as



**FIGURE 11.** latency comparison from the sensor to actuator for various applications (A)Hypotension (B)Cardiovascular (C)Arrhythmia (D)combined application.

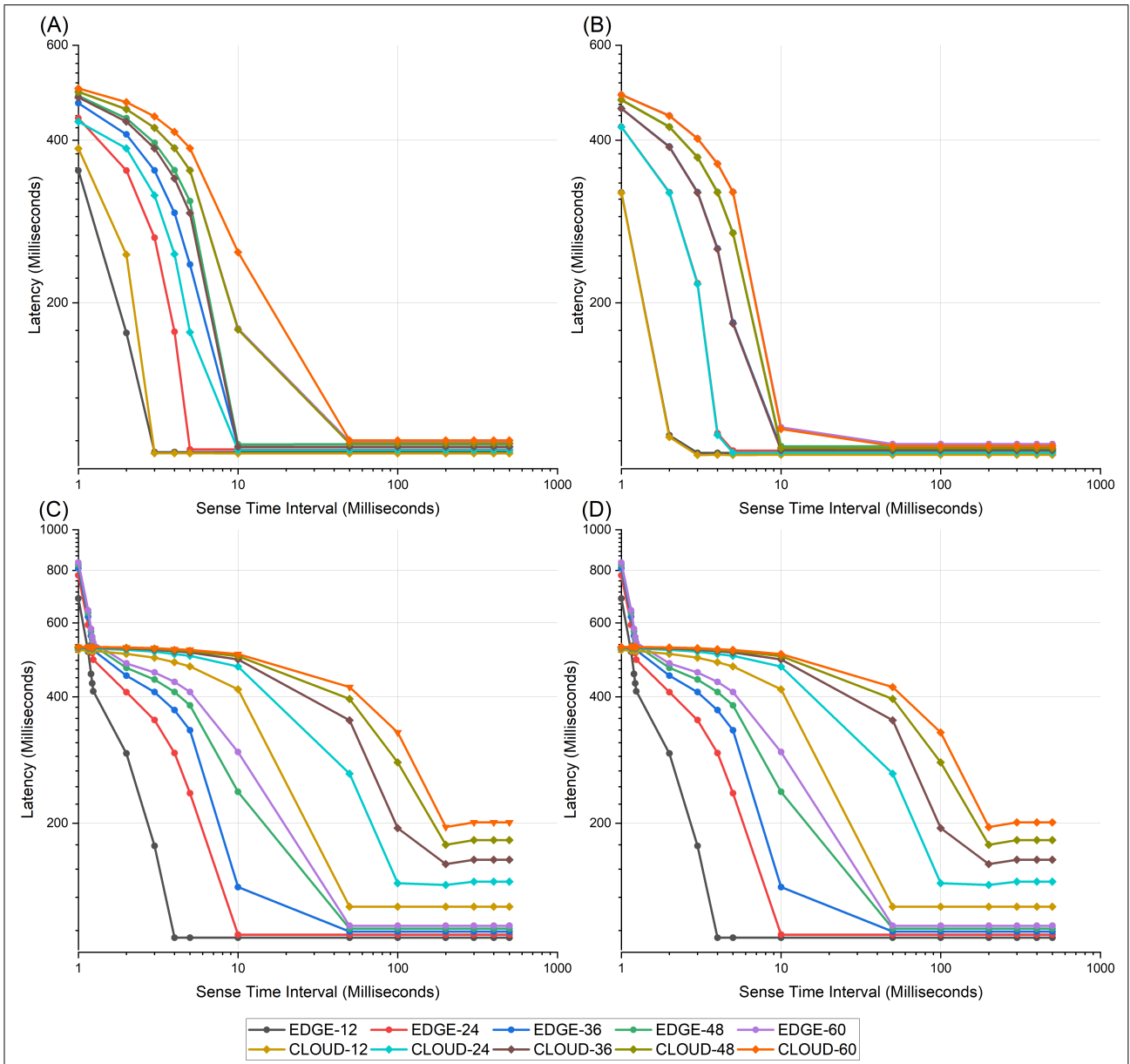
between sensors and the user interface, in the two distinct scenarios mentioned earlier in this section. From Table 7, it can be observed that latency for sensor to actuator in hypotension application, there is 70 times improvement with *Config-1* which is increased to 124 times with *config-5*. The same trend applies to the applications of cardiovascular disease, arrhythmia and combination of all.

Figures 11 also depicts the comparison of latency for various sense time intervals in both edge and cloud environments. The findings indicate a sense time interval ranging from 500 ms to 1 ms. The sensor-to-actuator latency increases gradually on decreasing the sense time interval. However, for edge computing, there is a sudden increase of latency on decreasing the sense time interval at a certain threshold point due to limited computing capability of the edge gateway. We observed that this inflection point changes on changing the compute power of the edge gateway.

For instance, in cases of hypotension and cardiovascular disease, latency remains unchanged due to the small data packet size or the minimal computation power needed, indicating no burden on the computing node. However, in the case of arrhythmia and combined applications, latency starts increasing at a sense time interval of approximately 5ms, due to the burden on the edge computing node.

Meanwhile, if the sense time is decreased to 0.5 ms, the latency of the combined application may be higher in the Edge environment compared to the Cloud environment.

Figures 12 also presents the comparison of latency for various sense time intervals, ranging from 500 ms to 1 ms, in both environments. The latency observed in the user-interface is nearly identical between the edge and cloud environments for hypotension and cardiovascular disease applications, as no significant compute burden is placed on edge computing. However, in the case of arrhythmia and



**FIGURE 12.** latency comparison from the sensor to a user interface for various applications (A)Hypotension (B)Cardiovascular (C)Arrhythmia(D)combined application.

combined applications, the sensor-to-user interface latency is higher in the edge environment compared to the cloud environment after particular sense time interval. This can be attributed to edge computing requiring more time to process data or make decisions, particularly with larger packet sizes.

From Table 8, it can be observed that latency for sensor to user-interface in hypotension application does not have significant change with any of the physical topology configurations. It is similar for other applications.

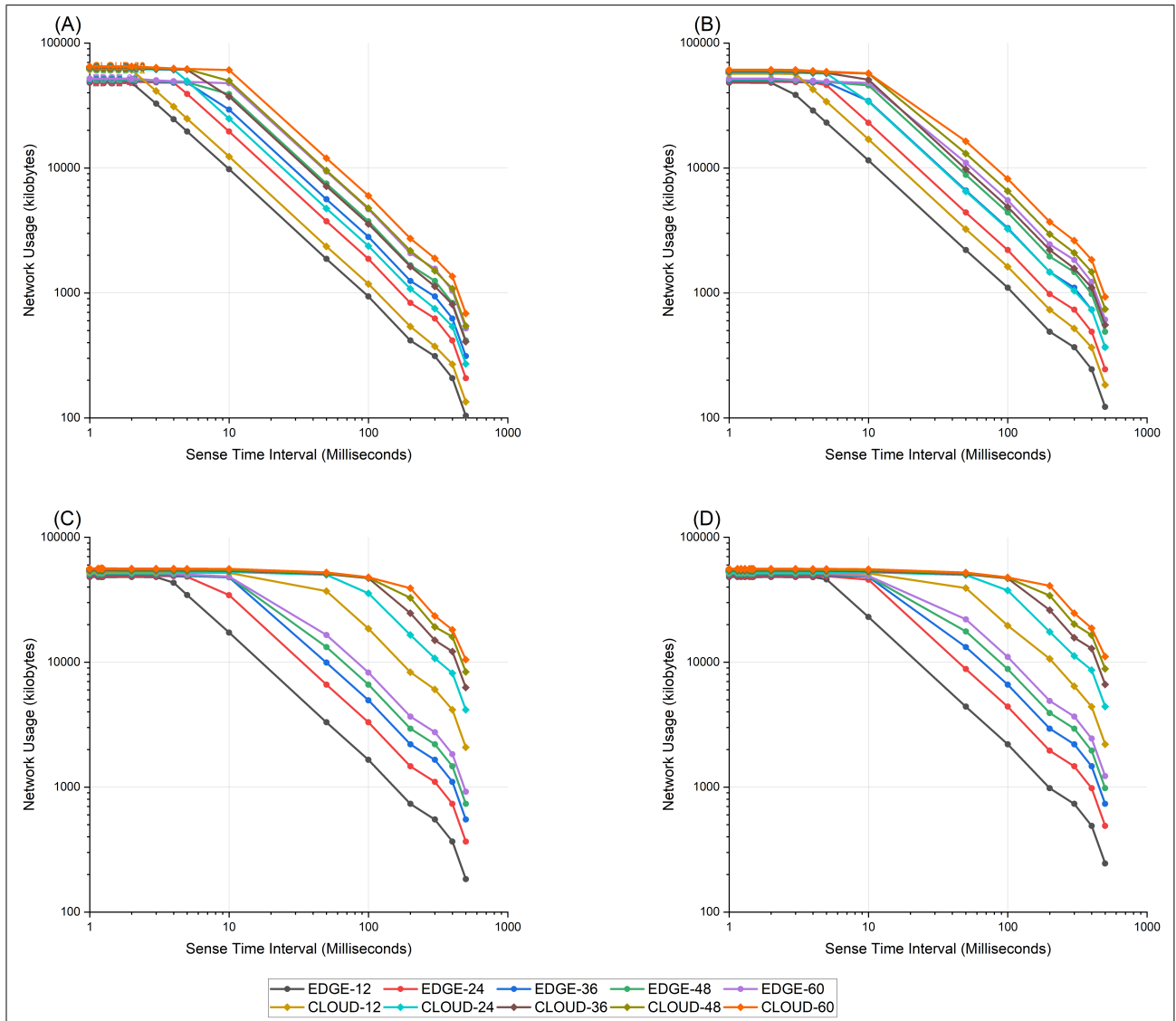
**B. ANALYSIS OF NETWORK USAGE**

When the workload on a cloud server increases, it often relies solely on cloud infrastructure, resulting in elevated network

**TABLE 7.** Simulation results for sensor to actuator latency.

Application	Hypotension	Cardiovascular	Arrhythmia	Combined
Config-1	71.52	73.48	77.07	63.13
Config-2	83.12	87.92	83.31	69.00
Config-3	94.44	98.79	87.77	73.33
Config-4	108.87	105.20	91.66	76.45
Config-5	124.97	109.83	94.51	78.71

usage due to the surge in visitors to the cloud server. This increased traffic can lead to a reduction in data transmission rates on the network. However, in the case of geographically dispersed servers, a dedicated Edge gateway is allocated to manage requests from specific sensor nodes. Consequently, network consumption decreases under these circumstances, but the transmission rate for the remaining network traffic sees an increase.



**FIGURE 13.** Comparing network usage for different applications, including (A) Hypotension, (B) Cardiovascular, (C) Arrhythmia, and (D) Combined application.

**TABLE 8.** Simulation results for sensor to user interface latency.

Application	Hypotension	Cardiovascular	Arrhythmia	Combined
Config-1	1.04	0.99	1.86	1.54
Config-2	1.10	0.99	1.54	1.36
Config-3	1.05	0.99	1.50	1.28
Config-4	1.08	0.99	1.48	1.32
Config-5	1.06	0.99	1.51	1.36

Equation 10, as cited from [6], is utilized to determine network utilization.

$$\text{Network usage} = \text{Latency} * \partial \tag{10}$$

where

$$\partial = \text{tupleNWSize}$$

We conducted simulations to demonstrate the efficiency of our proposed edge-based Clinical Decision Support System (CDSS). We explored multiple scenarios, involving both edge

and cloud setups, each with varying numbers of patients connected to the edge gateway and cloud server. Specifically, we established two edge gateways for evaluation. Each edge gateway is linked to the 12 sensor nodes. This configuration maintained a consistent number of edge gateways throughout the performance assessment. For the cloud-based scenario, we utilized a router to link the sensor nodes to the cloud.

Our primary focus was on calculating latency and network utilization. Figures 11 and 12 illustrate a comparison of latency in both the cloud and edge environments. The results reveal that as the number of sensor nodes increases, latency in the cloud environment increases significantly more than in the edge environment. This discrepancy occurs because, in the edge setup, each edge gateway exclusively manages data for the ward it is connected to, whereas in the cloud setup, the server processes data for all ICU patients, resulting in increased latency.



Figure 13 presents network utilization data for the edge gateway scenario, showing an increase in network utilization as the number of edge gateways and sensor nodes increases. In contrast, the cloud environment exhibits higher network utilization because it processes all sensor data simultaneously, leading to greater network demand. In the edge setup, various sensor nodes are connected to different edge gateways, with each edge gateway dedicated to a specific ICU ward, resulting in more efficient data processing.

Figure 13 also shows that the network utilization as compared between the cloud and edge architectures is similar for hypotension and cardiovascular disease across all sense time intervals. However, notable difference was observed for arrhythmia and combined applications, primarily because the amount of data transmitted to the cloud is consistently lower in the edge environment compared to the cloud environment for all sense time intervals.

From Table 9, there is no significant change in network usage in the hypotension application, but there is a 5-fold improvement for the arrhythmia application and a 4 times improvement for combination of all. This is due to the size of the data transmitted.

**TABLE 9. Simulation results for network usage.**

Application	Hypotension	Cardiovascular	Arrhythmia	Combined
Config-1	1.26	1.43	5.05	4.38
Config-2	1.27	1.36	4.57	3.42
Config-3	1.27	1.36	4.27	3.21
Config-4	1.27	1.34	4.01	3.02
Config-5	1.27	1.34	3.77	2.86

Our comprehensive experiments for both the edge-based and cloud-based approaches, taking into account latency and network utilization, demonstrate the effectiveness of the proposed edge-based architecture for vital monitoring systems.

The simulation results for both edge gateway-based and cloud-based implementations, considering latency and network usage, highlight the practicality and efficiency of the edge-based architecture for vital monitoring systems. Utilizing an edge gateway-based architecture for smart vital monitoring allows us to promptly detect abnormalities in patients' vital signs, predict future changes, and thereby reduce patient risk. Additionally, these findings underscore the potential of edge computing in IoT applications where rapid response times are essential. In summary, the low latency and minimal network usage of the edge-based architecture make it highly suitable for real-time applications and scenarios.

## VI. CONCLUSION

In recent years, edge computing has assumed an increasingly critical role, particularly in time-sensitive applications such as healthcare. The exponential growth of data generated by medical devices has amplified the demand for faster response times and disease prediction. Also, the data requirement and the compute algorithm required for the same varies for each disease/condition prediction. To address these challenges,

an IoT-based clinical decision support system has been proposed in this paper, which uses an edge-compute gateway architecture that can be configured on the fly for multiple data flow patterns and decision-making algorithms.

The proposed gateway was implemented using iFogSim. The simulation results show an average reduction of latency by 87.66 times compared to a cloud-based architecture. The latency is impacted by the packet size, the sense time interval, and the number of devices attached to the edge gateway. Similarly, network usage improves by 2.59 times as compared to the cloud-based architecture. The proposed gateway has been shown to support multiple applications simultaneously by configuring the gateway with respect to the buffer sizes needed for each application and the decision-making algorithm it uses. Looking ahead, this research can be extended to explore issues related to load balancing, offloading, and security in Edge Compute Gateway-based clinical decision support systems. The proposed gateway architecture can also be deployed for applications outside the healthcare domain.

## REFERENCES

- [1] R. T. Sutton, D. Pincock, D. C. Baumgart, D. C. Sadowski, R. N. Fedorak, and K. I. Kroeker, "An overview of clinical decision support systems: Benefits, risks, and strategies for success," *Npj Digit. Med.*, vol. 3, no. 1, pp. 1–10, Feb. 2020.
- [2] I. C. Jeong, D. Bychkov, and P. C. Seanson, "Wearable devices for precision medicine and health state monitoring," *IEEE Trans. Biomed. Eng.*, vol. 66, no. 5, pp. 1242–1258, May 2019.
- [3] A. Nikam, S. Bhandari, A. Mhaske, and S. Mantri, "Cardiovascular disease prediction using machine learning models," in *Proc. IEEE Pune Sect. Int. Conf. (PuneCon)*, Dec. 2020, pp. 22–27.
- [4] A. T. M. Wasylewicz and A. M. J. W. Scheepers-Hoeks, "Clinical decision support systems," in *Fundamentals of Clinical Data Science*, P. Kubben, et al., Eds. Springer, 2019, pp. 153–169.
- [5] M. P. Sundaramurthy, "IoT on healthcare using clinical decision support system," in *Diagnostic Applications of Health Intelligence and Surveillance Systems*. Hershey, PA, USA: IGI Global, 2021, pp. 259–280.
- [6] H. Gupta, A. V. Dastjerdi, S. K. Ghosh, and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.
- [7] S. Secinaro, D. Calandra, A. Secinaro, V. Muthurangu, and P. Biancone, "The role of artificial intelligence in healthcare: A structured literature review," *BMC Med. Informat. Decis. Making*, vol. 21, no. 1, pp. 1–23, Dec. 2021.
- [8] D. Li, "5G and intelligence medicine—How the next generation of wireless technology will reconstruct healthcare?" *Precis. Clin. Med.*, vol. 2, no. 4, pp. 205–208, Dec. 2019.
- [9] A. Ahad, M. Tahir, M. Aman Sheikh, K. I. Ahmed, A. Mughees, and A. Numani, "Technologies trend towards 5G network for smart health-care using IoT: A review," *Sensors*, vol. 20, no. 14, p. 4047, Jul. 2020.
- [10] M. N. Bhuiyan, M. M. Rahman, M. M. Billah, and D. Saha, "Internet of Things (IoT): A review of its enabling technologies in healthcare applications, standards protocols, security, and market opportunities," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10474–10498, Jul. 2021.
- [11] S. Selvaraj and S. Sundaravadhan, "Challenges and opportunities in IoT healthcare systems: A systematic review," *Social Netw. Appl. Sci.*, vol. 2, no. 1, pp. 1–8, Jan. 2020.
- [12] A. Gatouillat, Y. Badr, B. Massot, and E. Sejdic, "Internet of Medical Things: A review of recent contributions dealing with cyber-physical systems in medicine," *IEEE Internet Things J.*, vol. 5, no. 5, pp. 3810–3822, Oct. 2018.
- [13] N. Mani, A. Singh, and S. L. Nimmagadda, "An IoT guided healthcare monitoring system for managing real-time notifications by fog computing services," *Proc. Comput. Sci.*, vol. 167, pp. 850–859, Jan. 2020.

- [14] A. A. Mutlag, M. K. Abd Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems," *Future Gener. Comput. Syst.*, vol. 90, pp. 62–78, Jan. 2019.
- [15] J. A. Roth, M. Battagay, F. Juchler, J. E. Vogt, and A. F. Widmer, "Introduction to machine learning in digital healthcare epidemiology," *Infection Control Hospital Epidemiol.*, vol. 39, no. 12, pp. 1457–1462, Dec. 2018.
- [16] J. Wiens and E. S. Shenoy, "Machine learning for healthcare: On the verge of a major shift in healthcare epidemiology," *Clin. Infectious Diseases*, vol. 66, no. 1, pp. 149–153, Jan. 2018.
- [17] C. S. Kruse, A. Stein, H. Thomas, and H. Kaur, "The use of electronic health records to support population health: A systematic review of the literature," *J. Med. Syst.*, vol. 42, no. 11, pp. 1–16, Nov. 2018.
- [18] P. Campanella, E. Lovato, C. Marone, L. Fallacara, A. Mancuso, W. Ricciardi, and M. L. Specchia, "The impact of electronic health records on healthcare quality: A systematic review and meta-analysis," *Eur. J. Public Health*, vol. 26, no. 1, pp. 60–64, Feb. 2016.
- [19] L. Poissant, J. Pereira, R. Tamblin, and Y. Kawasumi, "The impact of electronic health records on time efficiency of physicians and nurses: A systematic review," *J. Amer. Med. Inform. Assoc.*, vol. 12, no. 5, pp. 505–516, May 2005.
- [20] F. A. Reegu, H. Abas, Y. Gulzar, Q. Xin, A. A. Alwan, A. Jabbari, R. G. Sonkamble, and R. A. Dziyauddin, "Blockchain-based framework for interoperable electronic health records for an improved healthcare system," *Sustainability*, vol. 15, no. 8, p. 6337, Apr. 2023.
- [21] P. S. Mathew, A. S. Pillai, and V. Palade, "Applications of IoT in healthcare," in *Cognitive Computing for Big Data Systems Over IoT*. Berlin, Germany: Springer, 2018, pp. 263–288.
- [22] L. C. K. Man, C. M. Na, and N. C. Kit, "IoT-based asset management system for healthcare-related industries," *Int. J. Eng. Bus. Manag.*, vol. 7, pp. 7–19, Jan. 2015.
- [23] Y. Wang, J. He, H. Zhao, Y.-H. Han, and X.-J. Huang, "Intelligent community medical service based on Internet of Things," *J. Interdiscipl. Math.*, vol. 21, no. 5, pp. 1121–1126, Jul. 2018.
- [24] N. L. Fitriyani, M. Syafrudin, G. Alfian, and J. Rhee, "HDPM: An effective heart disease prediction model for a clinical decision support system," *IEEE Access*, vol. 8, pp. 133034–133050, 2020.
- [25] D. Dias and J. Paulo Silva Cunha, "Wearable health devices—Vital sign monitoring, systems and technologies," *Sensors*, vol. 18, no. 8, p. 2414, Jul. 2018.
- [26] T. Ahrens, "The most important vital signs are not being measured," *Austral. Crit. Care*, vol. 21, no. 1, pp. 3–5, Feb. 2008.
- [27] M. Elliott and A. Coventry, "Critical care: The eight vital signs of patient monitoring," *Brit. J. Nursing*, vol. 21, no. 10, pp. 621–625, May 2012.
- [28] M. Kebe, R. Gadhafi, B. Mohammad, M. Sanduleanu, H. Saleh, and M. Al-Qutayri, "Human vital signs detection methods and potential using radars: A review," *Sensors*, vol. 20, no. 5, p. 1454, Mar. 2020.
- [29] J. Liu, Y. Chen, Y. Wang, X. Chen, J. Cheng, and J. Yang, "Monitoring vital signs and postures during sleep using WiFi signals," *IEEE Internet Things J.*, vol. 5, no. 3, pp. 2071–2084, Jun. 2018.
- [30] S. Bashir, A. A. Almazroi, S. Ashfaq, A. A. Almazroi, and F. H. Khan, "A knowledge-based clinical decision support system utilizing an intelligent ensemble voting scheme for improved cardiovascular disease prediction," *IEEE Access*, vol. 9, pp. 130805–130822, 2021.
- [31] A. A. Montgomery, "Evaluation of computer based clinical decision support system and risk chart for management of hypertension in primary care: Randomised controlled trial," *BMJ*, vol. 320, no. 7236, pp. 686–690, Mar. 2000.
- [32] N. Kaieski, C. A. da Costa, R. D. R. Righi, P. S. Lora, and B. Eskofier, "Application of artificial intelligence methods in vital signs analysis of hospitalized patients: A systematic literature review," *Appl. Soft Comput.*, vol. 96, Nov. 2020, Art. no. 106612.
- [33] J. G. Chester and J. L. Rudolph, "Vital signs in older patients: Age-related changes," *J. Amer. Med. Directors Assoc.*, vol. 12, no. 5, pp. 337–343, Jun. 2011.
- [34] N. Peiffer-Smadja, T. M. Rawson, R. Ahmad, A. Buchard, P. Georgiou, F.-X. Lescure, G. Birgand, and A. H. Holmes, "Machine learning for clinical decision support in infectious diseases: A narrative review of current applications," *Clin. Microbiology Infection*, vol. 26, no. 5, pp. 584–595, May 2020.
- [35] B. Prajapati, S. Parikh, and J. Patel, "An intelligent real time IoT based system (IRTBS) for monitoring ICU patient," in *Information and Communication Technology for Intelligent Systems*. Berlin, Germany: Springer, 2018, pp. 390–396.
- [36] A. Abdelgawad, K. Yelamarthi, and A. Khattab, "IoT-based health monitoring system for active and assisted living," in *Smart Objects and Technologies for Social Good* (Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering), vol. 195, O. Gaggi, P. Manzoni, C. Palazzi, A. Bujari, and J. Marquez-Barja, Eds. Cham, Switzerland: Springer, 2017.
- [37] M. Salem, A. Elkaseer, I. A. M. El-Maddah, K. Y. Youssef, S. G. Scholz, and H. K. Mohamed, "Non-invasive data acquisition and IoT solution for human vital signs monitoring: Applications, limitations and future prospects," *Sensors*, vol. 22, no. 17, p. 6625, Sep. 2022, doi: 10.3390/s22176625.
- [38] T. N. Gia, I. B. Dhaou, M. Ali, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Energy efficient fog-assisted IoT system for monitoring diabetic patients with cardiovascular disease," *Future Gener. Comput. Syst.*, vol. 93, pp. 198–211, Apr. 2019.
- [39] J. Pan and J. McElhannon, "Future edge cloud and edge computing for Internet of Things applications," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, Feb. 2018.
- [40] P. McEnroe, S. Wang, and M. Liyanage, "A survey on the convergence of edge computing and AI for UAVs: Opportunities and challenges," *IEEE Internet Things J.*, vol. 9, no. 17, pp. 15435–15459, Sep. 2022.
- [41] B. Chen, J. Wan, A. Celestii, D. Li, H. Abbas, and Q. Zhang, "Edge computing in IoT-based manufacturing," *IEEE Commun. Mag.*, vol. 56, no. 9, pp. 103–109, Sep. 2018.
- [42] J. Pérez, J. Díaz, J. Berrocal, R. López-Viana, and Á. González-Prieto, "Edge computing: A grounded theory study," *Computing*, vol. 104, no. 12, pp. 2711–2747, Dec. 2022.
- [43] O. Salman, I. Elhadj, A. Kayssi, and A. Chehab, "Edge computing enabling the Internet of Things," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, Dec. 2015, pp. 603–608.
- [44] M. Laroui, B. Nour, H. Mounqila, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities & future directions," *Comput. Commun.*, vol. 180, pp. 210–231, Dec. 2021.
- [45] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.
- [46] T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge computing in industrial Internet of Things: Architecture, advances and challenges," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2462–2488, 4th Quart., 2020.
- [47] P. K. Sharma, S. Rathore, Y.-S. Jeong, and J. H. Park, "SoftEdgeNet: SDN based energy-efficient distributed network architecture for edge computing," *IEEE Commun. Mag.*, vol. 56, no. 12, pp. 104–111, Dec. 2018.
- [48] S. Hamdan, M. Ayyash, and S. Almajali, "Edge-computing architectures for Internet of Things applications: A survey," *Sensors*, vol. 20, no. 22, p. 6441, Nov. 2020.
- [49] D. C. Nguyen, P. N. Pathirana, M. Ding, and A. Seneviratne, "Blockchain and edge computing for decentralized EMRs sharing in federated healthcare," in *Proc. IEEE Global Commun. Conf.*, Dec. 2020, pp. 1–6.
- [50] A. Singh and K. Chatterjee, "Edge computing based secure health monitoring framework for electronic healthcare system," *Cluster Comput.*, vol. 26, no. 2, pp. 1205–1220, Apr. 2023.
- [51] M. Prabhu and A. Hanumanthaiah, "Edge computing-enabled healthcare framework to provide telehealth services," in *Proc. Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*, Mar. 2022, pp. 349–353.
- [52] J. Li, J. Cai, F. Khan, A. U. Rehman, V. Balasubramaniam, J. Sun, and P. Venu, "A secured framework for SDN-based edge computing in IoT-enabled healthcare system," *IEEE Access*, vol. 8, pp. 135479–135490, 2020.
- [53] M. Z. Uddin, "A wearable sensor-based activity prediction system to facilitate edge computing in smart healthcare system," *J. Parallel Distrib. Comput.*, vol. 123, pp. 46–53, Jan. 2019.
- [54] A. A. Abdellatif, A. Mohamed, C. F. Chiasserini, M. Tlili, and A. Erbad, "Edge computing for smart health: Context-aware approaches, opportunities, and challenges," *IEEE Netw.*, vol. 33, no. 3, pp. 196–203, May 2019.
- [55] P. Dong, Z. Ning, M. S. Obaidat, X. Jiang, Y. Guo, X. Hu, B. Hu, and B. Sadoun, "Edge computing based healthcare systems: Enabling decentralized health monitoring in Internet of Medical Things," *IEEE Netw.*, vol. 34, no. 5, pp. 254–261, Sep. 2020.
- [56] H. Wang, J. Gong, Y. Zhuang, H. Shen, and J. Lach, "Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes," in *Proc. Int. Conf. Netw., Archit., Storage (NAS)*, Aug. 2017, pp. 1–2.

- [57] P. Sethi and S. Sarangi, "Internet of Things: Architectures, protocols, and applications," *J. Electr. Comput. Eng.*, pp. 1–25, 2017, doi: [10.1155/2017/9324035](https://doi.org/10.1155/2017/9324035).
- [58] M. El-hajj, H. Mousawi, and A. Fadlallah, "Analysis of lightweight cryptographic algorithms on IoT hardware platform," *Future Internet*, vol. 15, no. 2, p. 54, Jan. 2023.
- [59] R. D. Masram and J. Abraham, "Efficient selection of compression-encryption algorithms for securing data based on various parameters," *College Eng.*, Pune, 2014.
- [60] M. Ashouri, F. Lorig, P. Davidsson, and R. Spalazzese, "Edge computing simulators for IoT system design: An analysis of qualities and metrics," *Future Internet*, vol. 11, no. 11, p. 235, Nov. 2019.
- [61] R. Buyya and S. N. Srirama, *Fog and Edge Computing: Principles and Paradigms*. Hoboken, NJ, USA: Wiley, 2019.
- [62] E. Baccarelli, P. G. V. Naranjo, M. Scarpiniti, M. Shojafar, and J. H. Abawajy, "Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study," *IEEE Access*, vol. 5, pp. 9882–9910, 2017.
- [63] K. S. Awaisi, A. Abbas, M. Zareei, H. A. Khattak, M. U. S. Khan, M. Ali, I. U. Din, and S. Shah, "Towards a fog enabled efficient car parking architecture," *IEEE Access*, vol. 7, pp. 159100–159111, 2019.
- [64] S. Sharma, M. F. Hashmi, and P. T. Bhattacharya, "Hypotension," in *StatPearls [Internet]*. Treasure Island, FL, USA: StatPearls, Jan. 2024. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK499961/>
- [65] D. DeMers and D. Wachs, "Physiology, mean arterial pressure," in *StatPearls [Internet]*. Treasure Island, FL, USA: StatPearls, Jan. 2024. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK538226/>
- [66] K. A. Kumari, G. S. Sadasivam, D. Dharani, and M. Niranjnamurthy, *Edge Computing: Fundamentals, Advances and Applications* (Advances in Industry 4.0 and Machine Learning). Boca Raton, FL, USA: CRC Press, 2021.
- [67] S. Ulianova, "Cardiovascular disease dataset," Tech. Rep. [Online]. Available: <https://www.kaggle.com/datasets/sulianova/cardiovascular-disease-dataset>
- [68] A. K. Dwivedi, "Performance evaluation of different machine learning techniques for prediction of heart disease," *Neural Comput. Appl.*, vol. 29, no. 10, pp. 685–693, May 2018.
- [69] R. J. P. Princy, S. Parthasarathy, P. S. H. Jose, A. R. Lakshminarayanan, and S. Jeganathan, "Prediction of cardiac disease using supervised machine learning algorithms," in *Proc. 4th Int. Conf. Intell. Comput. Control Syst. (ICICCS)*, May 2020, pp. 570–575.
- [70] R. V. Arjunan, "ECG signal classification based on statistical features with SVM classification," *Int. J. Adv. Signal Image Sci.*, vol. 2, no. 1, pp. 5–10, Jun. 2016.



**RACHURI HARISH KUMAR** received the bachelor's degree in electronics and communication engineering from the Keshav Memorial Institute of Technology (affiliated with JNTU Hyderabad), and the master's degree in embedded system from the CVSR College of Engineering (affiliated with JNTU Hyderabad). He is currently a Research Scholar with Mahindra University, Hyderabad, India. With five years of prior experience as an IoT and Embedded Engineer, his current research interests include the future Internet of Things, computing architectures, machine learning, and next-generation wireless communication and networking.



**BHARGHAVA RAJARAM** received the M.S. degree (by Research) in VLSI and embedded systems from IIIT-Hyderabad, and the Ph.D. degree in computing systems architecture from The University of Edinburgh. He is currently an Associate Professor with the Electrical and Computer Engineering Department, École Centrale School of Engineering, Mahindra University. His research interests include the Internet of Things, computer architecture, and embedded system design.

• • •