*Pakistan Journal of Science (Vol. 72 No. 4 December, 2020)*

# DISTRIBUTED INTEROPERABILITY SOLUTIONS SMART HEALTHCARE SYSTEM FOR MULTI-PATIENT VITAL SIGNS MONITORING AND FORECAST OF CRITICAL ALERTS

A. Jaleel[1], M. Awais[2], S. Khaldoon[2], S. Shahid[2] and M. Shehzad[1,]

[1]Department of Computer Science, Rachna College of University of Engineering and Technology, Lahore, Pakistan
[2]Department of Computer Science, University of Engineering and Technology, Lahore, Pakistan
Corresponding author's E-mail: abduljaleel@uet.edu.pk

**ABSTRACT:** In current healthcare systems, doctors and paramedics have to individually observe the readings on attached devices to judge a patient's condition. A smart healthcare system may generate an opinion about the patient by compiling data from various vital sign monitoring devices. However, multi-vendor devices face the data interoperability problem because of the varying standards used. Current solutions rely on centralized cloud/fog-based servers for interoperability which is a barrier to real-time multi-patient monitoring. This research presents an Edge-computing based distributed interoperability framework for smart healthcare devices and presents a system that continuously monitors the patients' vital signs, ensemble the results for display in the nursing office, or in the doctor's wallet. A healthcare setup was emulated for testing the proposed solution. Results are compared to the centralized authority-based system, which showed that the proposed solution performed better in terms of response time, with the advantage of utilizing the local resources to achieve data interoperability. We used deep learning techniques to learn patients' critical situation from the vital signs monitoring database and predicted the critical situations to alert about the critical patients with 86% accuracy and 91% precision. Our model achieved a sensitivity of 90% and specificity of 72%. Hence a good overall performance has been achieved.

**Keywords:** Health Informatics, Vital Sign Monitoring, Healthcare-IoT, Distributed Interoperability, Edge Computing, Critical Alerts Prediction, Deep Learning.

*(Received 01.09.2020 Accepted 23.11.2020)*

## INTRODUCTION

Healthcare Informatics and the Internet of Things (IoT) has improved the healthcare systems and biomedical field (Majhi *et al*., 2019) and enabled the medical devices to connect with healthcare IT systems through online computer networks (Pulkkis *et al*., 2017). Vendor-specific patient monitoring systems (Siwicki, 2020) are available that facilitate the doctors and paramedics to monitor a patient remotely. Also, the integration of patient-centric and multi-vendor healthcare devices is possible in smart medical applications with Medical Devices Plug and Play (MD-PnP) (Julian and Sue, 2012). But, besides many other compatibility and interoperability problems, one challenge is the difference in devices' operated data formats (Scmidt, 2013; Manogaran *et al*., 2018; Monica, 2018).

Patient vital signs reflect essential body functions like heartbeat, breathing rate, temperature, and blood pressure (MedlinePlus, 2019). Patients' vital signs monitoring devices and IoT-enabled healthcare solutions support a number of open and proprietary data formats (Pennic, 2014; CapsuleTech, 2018; CyberNET, 2020). Different health data standards like Consolidated-Clinical Document Architecture, Direct secure messaging, and Fast Healthcare Interoperability Resources are proposed for device interoperability (Pennic, 2014); however, vendors don't really want to come together (Monica, 2018), and the problem of data incompatibility still is a bottleneck. Moreover, patients visit various health care providers, hospitals and medical facilities, and the patients' healthcare data are temporally and spatially scattered (Neal, 2011) which brings challenges, including data heterogeneity and interoperability (Azaria *et al*., 2016).

The interoperability of healthcare monitoring devices with heterogeneous data formats is currently performed through cloud-based services (Chen *et al*., 2016; Jabbar *et al*., 2017), and a data request faces intrinsic network delays and traffic congestion (Vilela *et al*., 2020), which is a bottleneck to the real-time access of patients' data (Qadri *et al*., 2020). Thus, in the Healthcare-IoT domain with distributed data-sensing environment, the current solutions face deficiency of real-time data sharing among the heterogeneous devices, demanding near-network translations through the use of Edge/Fog based solutions (Nair and Tanwar 2020).

This research aims to relieve the burden of continuously monitoring a patient's vital signs for critical health conditions. By ensemble of the several vital signs data, e.g., blood pressure, breathing, pulse and temperature, a healthcare system may generate an opinion about the patient condition and alarms can be generated for the critical observations. This work is based on the hypothesis that a device within the premises of a healthcare provider or a patient, having the capability of data format conversion with an excess of computing resources, can provide its services to other devices when required. The services of these capable devices are need to be acquired to augment data interoperability services at edge devices, which are conventionally provided by the cloud/fog-based infrastructures.

We proposed a distributed framework to provide data interoperability solutions for a smart healthcare system that resolves the conflicts and mismatches of data format through "Distributed Services Registration" and "Distributed Conversion Manager" processes. Instead of requesting the cloud for data translations, the proposed framework enables smart healthcare devices to share the patient's data through translation capable devices in the vicinity. We used deep learning techniques to learn patients' critical situation from the vital signs monitoring database and predicted the critical situations to alert about the critical patients.

## MATERIAL AND METHODS

We divided the proposed system into four layers of processing, as given in Figure 1. At the bottom layer, we have a patient under observation by the smart vital sign monitoring devices. These devices sent the vital signs data to a controller device at the Edge. The controller device availed the data format's translation capabilities of Distributed Interoperability Devices present on the Edge. To make it possible, this research presented a framework for the distributed interoperability of smart healthcare devices. The Edge controller device compiled the vital signs readings and sent to the 'Nursing Office Display' for remote monitoring of multi-patients on a single screen.

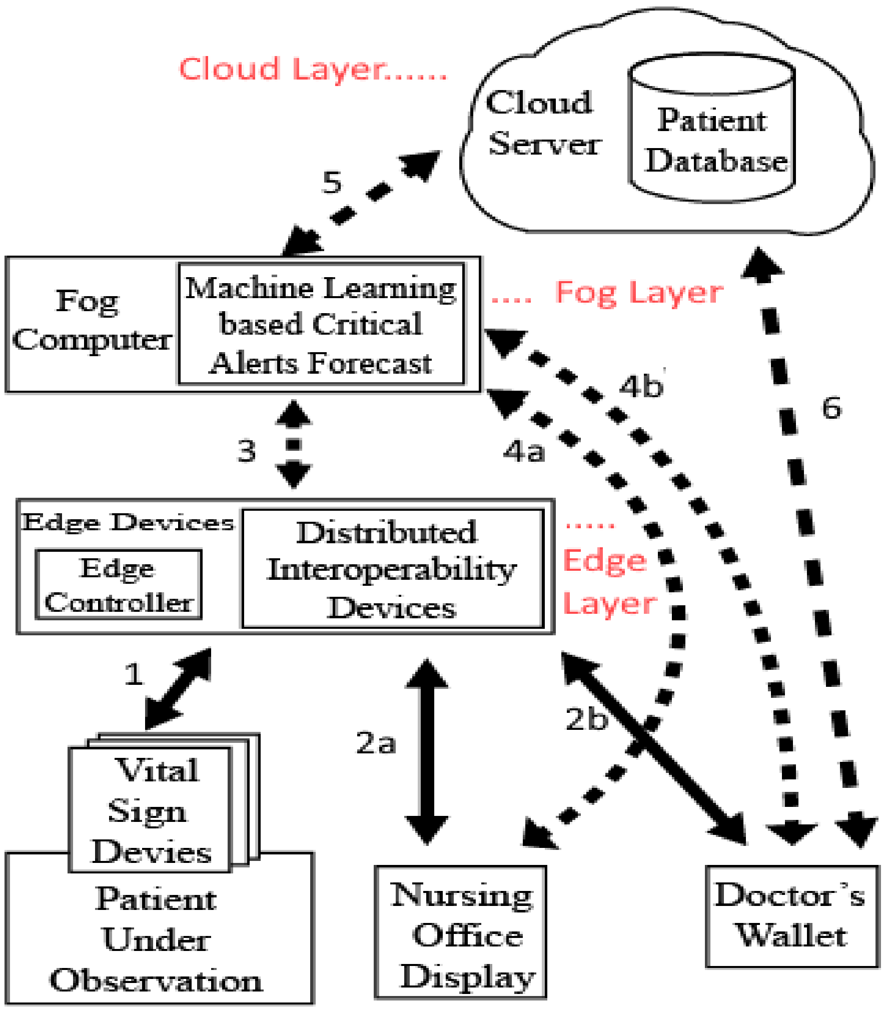


**Figure 1: Design Architecture of the Proposed System**

It also facilitated a doctor to access the patients' live condition through his wallet device, connected with the Edge-controller. The controller device (ESP32 controller) was programmed to generate an alarm on the connected displays if the compiled data of the patient's vital sign was crossing the given threshold. The Edge-controller also sent the compiled data of the patient's Vital Signs to the Fog-computer for further processing. We developed a deep-learning based critical alert forecasting module that took the compiled data and applied Convolutional Neural Network (CNN)-based deep learning algorithm to predict patient condition. CNN was trained for these forecasts with the patient vital signs monitoring database provided by the University of Queensland (Liu *et al*., 2012). We used the Jetson Nano Developer Kit from NVIDIA as our Fog-computer that produced the prediction about patient condition and sent these forecasts as alerts to the 'Nursing Office Display' and to the connected wallets of doctors. The developed system was capable of generating actual alerts and forecasting alerts. We maintained the patients' vital sign database on the cloud to store the information about critical patients so that a doctor could access the details at any time and prescribe the medication and other care-related action.

**Distributed Interoperability Framework:** We presented a distributed interoperability framework for data translations from capable devices, accessible to the hospital network and have spare computing resources. The proposed framework, shown in Figure 2, has facilitated to resolve the conflicts of various data formats.
.
It consisted of four layers, namely, Registration, Listener, Data interoperability, and Publish.

**Registration Layer:** It control the initialization of a transaction with token sharing based device authentication. This layer facilitated a healthcare provider or a smart healthcare device to get rights of interaction with a smart device by providing the key shared by the administrator. The registration layer supported two processes through its four modules. The first process was the Token Sharing handler that worked through *Token Request Client ()* and *Token Request Manager()* modules. The former executes in a Requesting Device, e.g., healthcare provider's wallet or any smart healthcare device that needed to connect with a patient's smart device. It sent the registration request to the Edge Controller to get Tokens and the URIs of patient's smart healthcare devices. The later executes inside the Edge Controller as a response side for authentication. When it received the registration request containing a valid key, it generated access-tokens for all the patient devices registered with the Edge Controller, mapped the devices in its device interaction table, and send the tokens and URIs od patient devices to the requesting device.

The second process handled by the registration layer was Distributed Services Registration that worked through its *Registration Controller ()* and *Registration Response ()* modules. The former executes in Edge Controller that broadcasted a request for registration of the nearby capable devices that have spare computing resources.

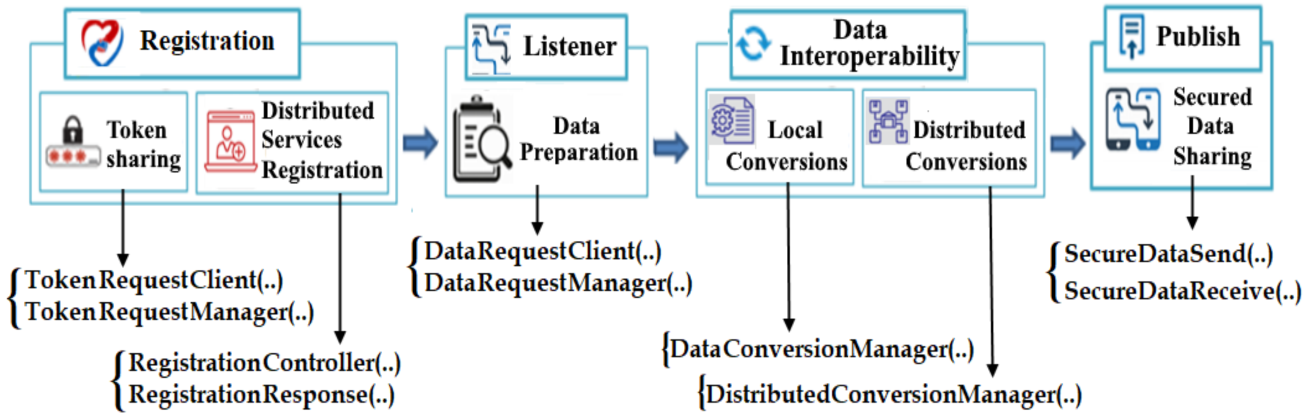


**Figure 2: Proposed Distributed Framework for Interoperability of smart healthcare Devices.**

The broadcast request asks if any capable device can convert the data from Format-X to Format-Y, denoting the current format and required format. The Edge Controller maintained a register for listing the capable devices indexed by the tuple (Format-X to Format-Y) as translation capability. An algorithm for the registration controller process is given in Algorithm 1.

**Algorithm 1: Registration Controller**

```
Input: Current Format, Required Format
Output: Capable Devices List
Start
A: Registration Broadcast
Broadcast Request (Current Format, Required Format)
devices = listen For Results()
```

```
if(!devicesis Empty())
 foreach D in devices
  if(D. Authenticated () && D.HOP count<Threshold
&&                 Already Exist==False) then
     capable Devices. add(D.URI, D. pass Key,
        D. hop Count, Current Format, Required Format)
B: Registration Renew
for each D in this. Capable Devices
  Thread. Timer(2).send Registration Renew (D.URI,
                          D. pass Key)
  if(D.reply() && D.HOP count<Threshold &&
                    Already Exist==True) then
      Capable Devices. update(D.URI)
  else
      capable Devices. Remove (D.URI)
End
```

The later executes inside the smart healthcare devices that support multiple data formats, have spare computing resources, and are capable of data conversion for other devices. On receiving a broadcast registration request from Edge Controller, smart devices responded with their URI, communication token and conversion capability. We present the working of the registration response module as Algorithm 2.

**Algorithm 2: Registration Response**

```
Input: Registration broadcasted or renewal request
received
Output: URI, PassKey, hopCount,
Start
CF= Request.getCurrentFormat()
RF= Request.getRequiredFormot()
if (this.canconvert (CF to RF) )
   cpbleList.add(this.URI, this.generatePassKey(),
                 d.getHopCount(), CF, RF)
foreach D in capableDevices
 if(D. canconvert (CF to RF)
 cpbleList.add(D.URI, D.PassKey(), d.HopCount(), CF,
RF)
End
```

*To ensures a maximum probability of getting a translation capable device, we programmed the capable devices to maintain a register of translation capable devices available in their vicinity.*

**Listener Layer:** After successful registration, the listener layer handles the data requests made from authentic devices. The *Data Request Client ()* was executed in the requesting devices as the client-side of the listener layer. The request for healthcare data was made using patient device URI and the token received from the Edge Controller. The *Data Request Manager ()* was executed in the patient's smart healthcare devices as the listener layer's response side. It validated the token and interpreted the data request to determine what data and in which format it was requested. If the device contained the data in the requested format, the data was published to the requesting device. However, if the requested data format was different, it forwarded the data to the data interoperability layer with a conversion request. However, if the token was not valid, then the request for data sharing was denied.

**Data Interoperability Layer:** This layer receives the data translation request to convert it into a requester's understandable format. The working of this layer is divided into two processes, namely, Local Conversions and Distributed Conversions. In the former process, Data Conversion Manager () was executed in the data device that made these devices capable of converting the data from the existing format to the required format by implementing the techniques given in previous researches (Kim *et al.*, 2009; Zhang *et al.*, 2018; Latif *et al.*, 2015; Zaima *et al.*, 2009; Mezei *et al.*, 2018; Ma *et al.*, 2018; Sonsilphong *et al.*, 2016) The data conversion process, we applied in this work is depicted in Figure 3(a). The data format mapping model for JSON, XML and text is depicted in Figure 3(b). The mapping functions were implemented using Python libraries and modules to translate the data from one format to another. The response to the data request was made by converting and sending the data to the requesting device. However, the data was transferred to the Distributed Conversions process if the device could not convert the data into the requested format.

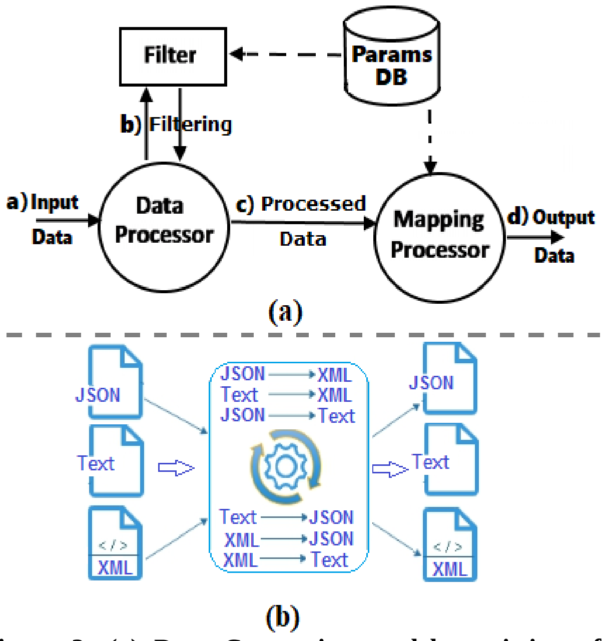


**Figure 3: (a) Data Conversion model consisting of data sensing, data pre-processing, filtering, and then Format Mapping, (b) A data format mapping model for JSON, XML and text**

When a device couldn't convert the data into a requested format and need help from an external device, it activated the Distributed Conversion Manager () by sending the data, current format, required format, requesting device URI, destination device URI, and Token key. This module got a list of conversion capable devices in the network from Edge-Controller. The controller device responded with a list of capable devices sorted priority wise on the base of the CPU-load factor and hop-count. The request was embedded with the payload and forwarded to an available device. The requesting device was intimated with the URI of a conversion device to receive data from that device. The capable device converted the data into the required data format through its data conversion manager and published the data directly to the requesting device. The algorithm for the module is given as Algorithm 3.

**Publish layer:** This layer provides methods for secure sending and receiving of data between smart healthcare devices. The Secure Data Send () module was executed in the smart healthcare devices to send data to a requesting device (a healthcare provider's wallet, a patient wallet, or other smart healthcare devices). After generating the healthcare data into a required format, the device called the secure data send method which applied encryption using the patient public key and added a checksum to the data with the SHA256 generator. HMAC/digital signature was added to data, which showed that the data was coming from the valid user and had not tampered. The data message was then transferred to the receiving side using an SSL/TLS (Cloudflare, 2019) handshake. The Secure Data Receive () module was executed in the data requesting devices to receive data from the patient's smart healthcare devices. The data message contained a digital signature, a checksum, and is encrypted with the patient public key. This module validated the data source and decrypted the data using the patient key.

Edge-Controller broadcasted the conversion request with the current format and the required format and capable devices that can convert, responded with URI, hop-count, CPU-load factor, and token..

**Algorithm 3: Distributed Conversion Manager**

```
Input: Message Type, Data, Current Format, Required
Format, RD.URI, DD.URI, PK
//RD: Requesting Device DD: Destination Device, PK:
Patient Key
Output: forward data for conversion & pulishing to
capable device
Start
if (Message-Type == "Conversion Request") then
Success = False
Capable Device List [URI,TOKEN] = Administering
Device. Get Conversion Devices List()
for each device in capable Device List
 if (Ping device) then //device available
   if (device. Can Convert (CurrentFormat,
                  Required Format))then
      Ack = device. convert & send (Data, Current Format,
              Required Format, RD.URI, PK, Token)
     if (Ack=="done")
       Success = True;
       return RD. revieve Data From (device. URI)
if(!Success)
   return "data not available in required format"
else // (invalid Request)
   return "Invalid Request Type"
End
```

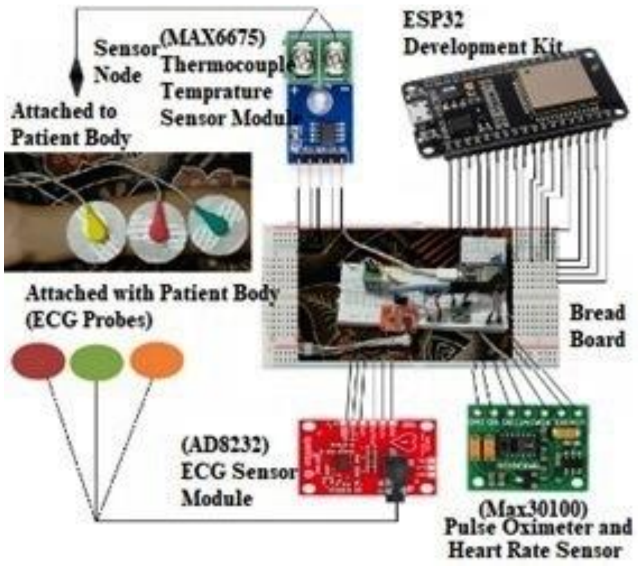
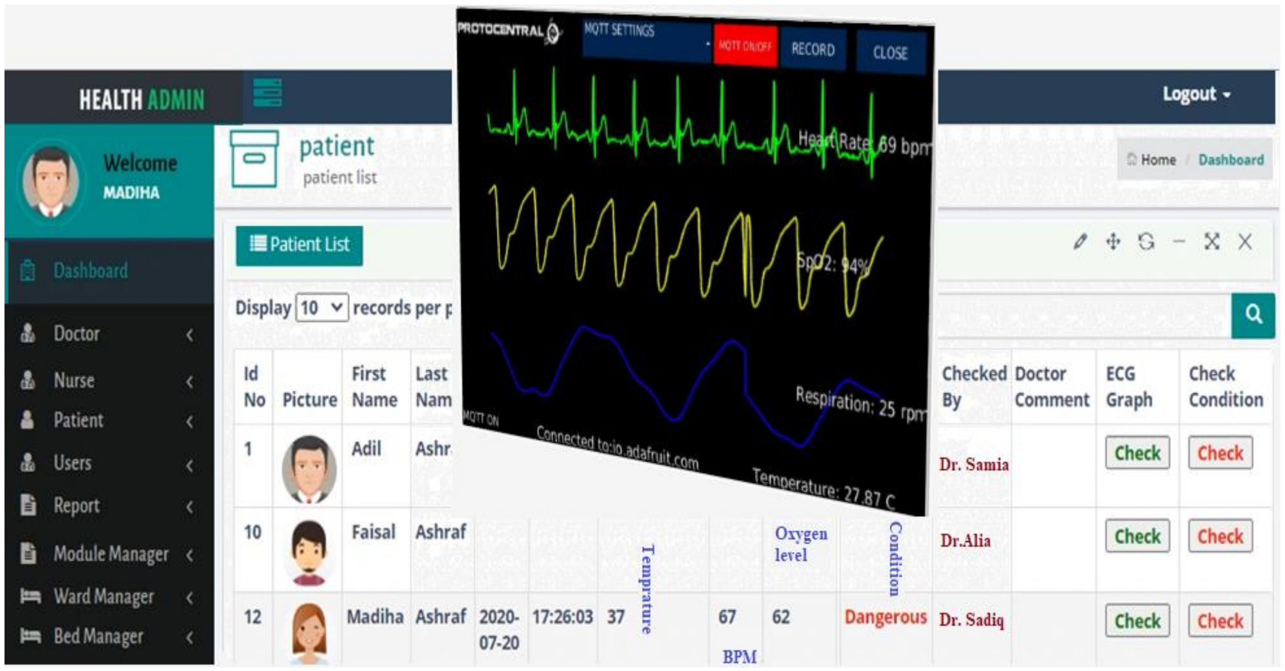

Sensor Node
Attached to Patient Body
(MAX6675) Thermocouple Temprature Sensor Module
ESP32 Development Kit
Attached with Patient Body (ECG Probes)
Bread Board
(AD8232) ECG Sensor Module
(Max30100) Pulse Oximeter and Heart Rate Sensor

**Figure 4: Experimental Setup for Patient Vital Sign Sensing System.**

**Experimental Setup:** We developed an IoT based Health Monitoring System for web and android platforms to achieve continuous monitoring of critical patients. The web-based dashboard was developed using Laravel Framework 5.8 with PHP 7.1.3 and the android application was developed in Android studio 3.5. The system stored the patient records in SQL real-time database at the cloud server that was used by web-based panel and android application. We designed an experimental setup for simulating the Intensive Healthcare Unit. The hardware modules were implemented using Arduino IDE and MQTT protocols. The setup was designed using ESP32 Controller and vital sign sensors including Temperature Sensor (MAX6675), ECG Sensor (AD8232) and Pulse Oximeter Sensor (MAX30100). A snapshot of the patient vital sign sensing system is depicted in Figure 4, whereas a portray of the working system is given as Figure 5. We also prepared software devices to represent different vendors' manufacturing with varying formats of data. The devices emulated for the experimentation were blood pressure monitoring (BPM), glucose level monitoring (GLM), ventilators for oxygen level flow (OLF) control, and heart-beat rate (HBR) monitor. Each of these devices was executed on a Raspberry Pi module. Apache JMeter 5.3 (Apache, 2020) was used for stress testing and analysis of test results. An automated system was designed to overcome the problems of overdosing, as it automatically adjusts the medicine input according to the patient readings. The controller device requested the readings from the patient devices and adjusted the dosage, flow, and speed of the infusion pump accordingly. For load tests, further IoT devices were programmed as soft devices in Java, and the system is connected using 100Mbps LAN.



| Id No | Picture | First Name | Last Nam |  |  |  | Temprature | BPM | Oxygen level | Condition | Checked By | Doctor Comment | ECG Graph | Check Condition |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 |  | Adil | Ashr |  |  |  |  |  |  |  | Dr. Samia |  | Check | Check |
| 10 |  | Faisal | Ashraf |  |  |  |  |  |  |  | Dr.Alia |  | Check | Check |
| 12 |  | Madiha | Ashraf | 2020-07-20 | 17:26:03 | 37 |  | 67 | 62 | Dangerous | Dr. Sadiq |  | Check | Check |

**Figure 5: A snapshot of the system developed for Multi-Patient Vital Sign Monitoring at Singel Screen**

## RESULTS AND DISCUSSION

To assess our proposed method, we set up two environments (a centralized Health IoT setup, and a distributed Health IoT setup) for device communication in the case study.

**Centralized Setup:** In this setup, smart healthcare devices were set up to take readings of healthcare data in different data formats like JSON, XML, Text, which are then sent to the server for storage. The server was installed with the data conversion manager module. So it was capable of responding to a request from any device by sending the data in the requested format. The data was received at the healthcare provider end after conversions at the cloud, so the problem of format mismatch was handled.

**Distributed Setup:** The second environment for devices' communication was designed with the proposed distributed framework. Here, the devices communicated directly with each other to translate and share the patient live monitoring data. The distributed framework based setup was divided into two scenarios.

***First Scenario:*** A smart healthcare device (having required data) implemented all layers of the proposed framework and was capable of converting and sharing healthcare records in the requested format. The healthcare provider wallet device was connected with the administering device (Edge Controller) using the patient key to get the URIs and tokens of the patient's smart healthcare devices. The healthcare provider then requested data from patient healthcare devices in the required format.

***Second Scenario:*** A Vital Sign Monitoring device had low processing capability or unable to convert the data in the requested format. It used the data format conversion capability of a nearby device available in the smart healthcare network. In this case, a 'distributed services registration controller' was installed in the Edge Controller to maintain connectivity parameters (URI, passkey, hop-count) of conversion-capable devices available in the smart healthcare network. The URIs, passkeys, and hop-counts were configured through the system administrator graphical user interface.

For experimental purposes, a data requesting interface was developed using HTML and JavaScript that acted as a data requesting device to get patient's healthcare data in XML, JSON, or Text format. We performed experimentation with an increasing number of software based (virtual) smart healthcare devices (10, 50, 100, 500, and 1000) to compare the average response time for different types of requests made to the centralized environment setup and to the distributed environment setup. The experimental configurations were as follows.

- In each slab of the devices, 70% were prepared as legacy devices that are unable to translate the data formats and need services from nearby capable devices.
- A 30% of the devices were prepared as translation capable devices, out of which 10% supports XML-to-JSON and vice versa, 10% supports XML-to-Text and vice versa, and the remaining 10% supports JSON-to-Text and vice versa.
- A device request was generated by selecting a random device on each clock tick, whereas the device's need for data was statically programmed.

To log the results, an experiment for each of the device's slab was executed for a duration of 100 seconds. The results of the experiments are given in Table 1. It shows that the central server took more time to provide a response to the requests when the number of data devices requesting increased.

**Table 1: AVERAGE RESPONSE TIME for Centralized Setup and Distributed Setup.**

| No. of Devices | Average Response Time (Centralized Setup) | Average Response Time (Distributed Setup) |
|---|---|---|
| 10 | 0.4250 sec | 0.392 sec |
| 50 | 0.5414 sec | 0.456 sec |
| 100 | 1.0854 sec | 0.470 sec |
| 500 | 1.6076 sec | 0.484 sec |
| 1000 | 1.7243 sec | 0.5014 sec |

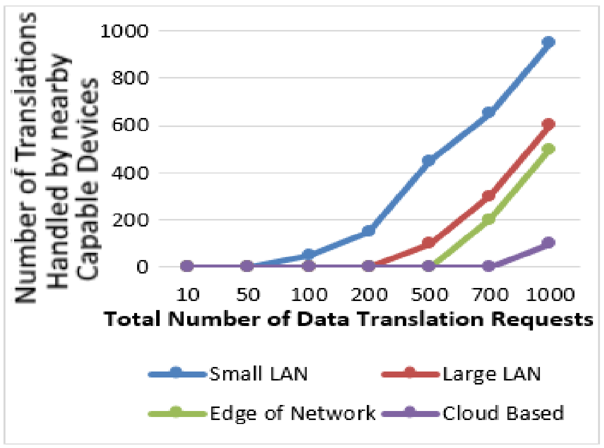
Whereas in the case of our proposed distributed framework based setup, the data translations were performed through local devices, and hence an increase in the number of devices has a trivial effect over the average response time.

We performed another experimentation to log the network traffic load generated because of the distributed conversion to compare the proposed system with the existing centralized system. In the case of a centralized system, the translation tasks were to be performed by the cloud. For distributed setup, we categorized the task's processing in the four categories with details as given in Table 2. In the Small LAN based Intensive Care Unit, the capable devices were able to process a total of 50 data interoperability requests at a time. A scenario was generated where an oxygen flow controller was asking for data of blood pressure and pulse rate of a patient under intensive care. In Large LAN based Hospital Network, a high processing device (X-ray machine) was added with the distributed registration and translation service. In the WAN based Edge of Network Translator, a scenario was emulated for a doctor's interaction with the devices installed on patients' bed. The data format conversion requests from patient's devices were handled through the edge of the network device installed at the hospital network. The system forwarded the requests to the cloud for translation when it crossed the local capability threshold. In cloud-based translations, the patient's data translations were performed through the cloud-based server.

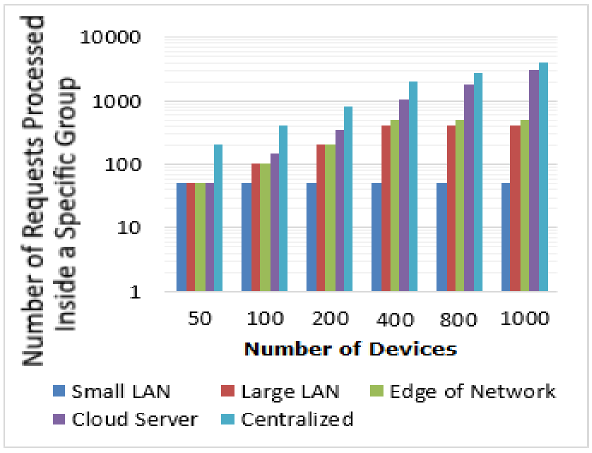
We generated an increasing number of data translation requests (1000 max), for centralized setup and for each of the four scenarios discussed above. We logged the number of translation requests and where each handled by capable devices. The results are shown in Figure 6(a) for 10, 50, 100, 200, 500, 700, and 1000 number of translation requests. We also logged the number of requests handled inside a specific group with the distributed setup and compared it with the centralized setup, Figure 6(b). It shows that the data transfer to the external network is less in the proposed framework-based setup than the existing cloud-based solutions.

**Table 2: Configurations to handle distributed conversion requests.**

| | Number of Devices | Conversion Capable Devices | Max. possible Interoperability Requests |
|---|---|---|---|
| *Small LAN (ICU)* | *50* | *5* | *50* |
| *Large LAN (Hospital)* | *200* | *20* | *200 local + 200 by X-Ray Machine* |
| *WAN/ Edge of Network* | *400* | *25* | *400 by Edge Controller* |
| *Cloud-based* | *1000* | *30* | *1000 by cloud* |



**(a)**



**(b)**

**Figure 6: Experimentation Results (a) Data Format conversion requests handled by conversion capable devices in the network, (b) A comparison of Data Format conversion requests processed by centralized versus proposed distributed framework based devices' communication.**

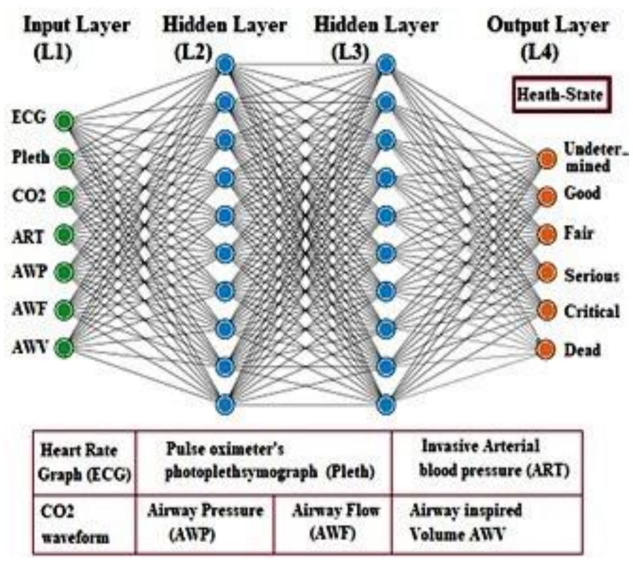


**Figure 7: Fully Connected Convolutional Neural Network used for training and testing**

**FORECASTING FOR CRITICAL ALERTS:** We applied the Convolutional Neural Network (CNN) for health state prediction and forecast of a patient under observation. At the input layer, CNN was taking the reading of seven vital signs, as depicted in Figure 7. CNN was designed to work with two hidden layers and six output classes which are given in Table 3 as health states defined by American Hospital (Wikipedia, 2019). We trained our CNN model with the vital signs database of critical patients provided by the University of Queensland (Liu, 2012). The database was containing vital sign reading of 32 cases (Patients). It consisted of 25 general anesthetics, out of which 20 were with an endotracheal tube, 5 with a laryngeal mask airway, 3 patients were with spinal anesthetics, and 4 sedation cases. Electrocardiography, pulse oximetry and noninvasive arterial blood pressure monitoring were used in all cases, while other monitors were used at the anesthesiologists' discretion. We used 70% of the data for trainings and remaining 30% was used for testing.

**Table 3: Vital Signs based patients' Health-states (American Hospital Association).**

| |
|---|
| **Undetermined:** Patient awaiting physician and/or assessment. |
| **Good:** Vital signs are stable and within normal limits. Patient is conscious and comfortable. Indicators are excellent. |
| **Fair:** Vital signs are stable and within normal limits. Patient is conscious, but may be uncomfortable. Indicators are favorable. |
| **Serious:** Vital signs may be unstable and not within normal limits. Patient is seriously ill. Indicators are questionable. |
| **Critical:** Vital signs are unstable and not within normal limits. Patient may be unconscious. Indicators are unfavorable. |
| **Dead:** Vital signs have ceased. Patient has died. |

The performance of the prediction and forecasting module was evaluated for the case 1 of dataset by logging the results as a confusion matrix which are presented in Table 4. For performance analysis of our proposed CNN-based model of patient health-condition prediction and forecasting, the values for Accuracy, Precision, Recall, and F1-measure were calculated.

**Table 4: Confusion Matrix.**

| | **Predictive Positive** | **Predictive Negative** | |
|---|---|---|---|
| **Actual Positive** | True Positives 45780 | False Negatives 4535 | **Sensitivity** $\frac{TP}{\mathrm{TP}+\mathrm{FN}}$ = 0.90 |
| **Actual Negative** | False Positives 4520 | True Negatives 11565 | **Specificity** $\frac{TN}{\mathrm{TN}+\mathrm{FP}}$ =0.72 |
| | **Precision** $\frac{TP}{TP+FP}$ = 0.91 | **F1 Score** $\frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}$ = 0.91 | **Accuracy** $\frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+TN+FP+\mathrm{FN}}$ = 0.86 |

Our CNN-model got 86% accuracy and 91% precision which are pretty much acceptable. The sensitivity was measured to be 90% and specificity was 72%. which are good. The results showed that a good overall performance has been achieved. We are actively working on further improvements.

**Conclusion:** To provide data interoperability among heterogeneous smart healthcare devices, we have proposed a framework in which Health IoT devices interact using a distributed network. The experiments were performed to prove the efficacy of the proposed distributed system in response time and data traffic. The research has enabled data-compatibility among smart healthcare devices for real-time data sensing, especially useful for monitoring ICU patients. This research has reduced the cost of health care and helped a healthcare setup to improve the treatment process. We presented a remote health monitoring system along with an alarm to alert critical health states. It will help medical staff shortage, reduces visits to the patient regularly, and checks patient condition remotely. In future, this work may be extended to deploy a nation-wide healthcare system for improving and managing the health status of nationals.

## REFERENCES

Apache., (2020). Apache jmeter - download apache jmeter. https://jmeter.apache.org/download_jmeter.cgi. (Accessed on 01/01/2020).

Azaria, A., Ekblaw, A., Vieira, T. and Lippman, A., (2016). August. Medrec: Using blockchain for medical data access and permission management. In 2016 2nd International Conference on Open and Big Data (OBD) (pp. 25-30). IEEE.

CapsuleTech., (2018). Compatibility of medical device data with hospital information systems (white paper). Technical report, CapsuleTech.com North America. URL https://capsuletech.com/wp-content/uploads/2019/08/Capsule_Whitepaper-CompatibilityDeviceData_082119.pdf. (Accessed on 07/19/2020).

Chen, S.W., Chiang, D.L., Liu, C.H., Chen, T.S., Lai, F., Wang, H. and Wei, W., (2016). Confidentiality protection of digital health records in cloud computing. Journal of medical systems, 40(5), p.124.

Cloudflare., (2019). What is ssl (secure sockets layer)?https://www.cloudflare.com/learning/ssl/what-is-ssl/. (Accessed on 05/13/2019).

CyberNET., (2020). A few problems medical professionals face with ehr compatibility– cybernet blog. https://www.cybernetman.com/blog/problems-medical-professionals-face-ehr-compatibility/, (Accessed on 26/04/2020).

Jabbar, S., Ullah, F., Khalid, S., Khan, M. and Han, K., (2017). Semantic interoperability in heterogeneous IoT infrastructure for healthcare. Wireless Communications and Mobile Computing, 2017.

Jaleel, A., Mahmood, T., Hassan, M.A., Bano, G. and Khurshid, S.K., (2020). Towards Medical Data Interoperability Through Collaboration of Healthcare Devices. IEEE Access, 8, pp.132302-132319.

Julian, G. and W. Sue., (2012). Medical device plug-and play (mdpnp) interoperability program (white paper). Technical report, URL http://www.mdpnp.org/uploads/Oct12_MD_PnP_White_Paper.pdf.Kshemkalyani, A.D. and

Singhal, M., 2011. Distributed computing: principles, algorithms, and systems. Cambridge University Press.

Kim, W., Lim, S., Ahn, J., Nah, J. and Kim, N., (2010). Integration of IEEE 1451 and HL7 exchanging information for patients' sensor data. Journal of medical systems, 34(6), pp.1033-1041.

Latif, S., Varaich, Z.A., Ali, M.A., Khan, M.A. and Ayyaz, M.N., (2015). Real-time health data acquisition and geospatial monitoring: A visual analytics approach. In 2015 International Conference on Open Source Systems & Technologies (ICOSST) (pp. 146-150). IEEE.

Liu, D., Gorges, M. and Jenkins, S.A., (2012). University of Queensland vital signs dataset: development of an accessible repository of anesthesia patient monitoring data for research. Anesthesia & Analgesia, 114(3), pp.584-589.

Ma, X., Wang, Z., Zhou, S., Wen, H. and Zhang, Y., (2018). Intelligent healthcare systems assisted by data analytics and mobile computing. In 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC) (pp. 1317-1322). IEEE.

Majhi, V., Paul, S. and Jain, R., (2019). Bioinformatics for healthcare applications. In 2019 Amity international conference on artificial intelligence (AICAI) (pp. 204-207). IEEE.

Manogaran, G., Chilamkurti, N. and Hsu, C.H., (2018). Emerging trends, issues, and challenges in Internet of Medical Things and wireless networks. Personal and Ubiquitous Computing, 22(5-6), pp.879-882.

MedlinePlus. (2019). Vital signs: Medlineplus medical encyclopedia. https://medlineplus.gov/ency/article/002341.htm,. (Accessed on09/27/2019).

Mezei, G., Somogyi, F.A. and Farkas, K., (2018). The Dynamic Sensor Data Description and Data Format Conversion Language. In ICSOFT (pp. 372-380).

Monica, K. (2018). How health data standards support healthcare interoperability. https://ehrintelligence.com/features/how-health-data-standards-support-healthcare-interoperability, (Accessed on 05/11/2019).

Nair, A., and Tanwar, S., (2020), Fog Computing Architectures and Frameworks for Healthcare 4.0. In Fog Computing for Healthcare 4.0 Environments (pp. 55-78). Springer, Cham.

Neal, D., (2011). Choosing an electronic health records system: Professional liability considerations. Innovations in Clinical Neuroscience, 8(6), p.43

Pennic, J. (2014). In-depth: A guide to healthcare data formats. https://hitconsultant.net/2014/04/28/in-depth-a-guide-to-healthcare-data-formats/#.Xrj1E2gvPIU,.(Accessed on 06/11/2019).

Pulkkis, G., Karlsson, J., Westerlund, M. and Tana, J., (2017), August. Secure and reliable Internet of Things systems for healthcare. In 2017 IEEE 5th international conference on future internet of things and cloud (FiCloud) (pp. 169-176). IEEE.

Qadri, Y.A., Nauman, A., Zikria, Y.B., Vasilakos, A.V. and Kim, S.W., (2020). The Future of Healthcare Internet of Things: A Survey of Emerging Technologies. IEEE Communications Surveys & Tutorials, 22(2), pp.1121-1167.

Scmidt, B. (2013). Medical device interoperability: A 'wicked problem' of our time-patient safety & quality healthcare. https://www.psqh.com/analysis/medical-device-interoperability-a-wicked-problem-of. (Accessed on 05/06/2019).

Siwicki, B. (2020). Updated: A guide to connected health device and remote patient monitoring vendors | healthcare it news. https://www.healthcareitnews.com/news/guide-connected-health-device-and-remote-patient-monitoring-2020. (Accessed on 09/27/2020).

Sonsilphong, S., Arch‐int, N., Arch‐int, S. and Pattarapongsin, C., (2016). A semantic interoperability approach to health‐care data: Resolving data‐level conflicts. Expert Systems, 33(6), pp.531-547.

Vilela, P.H., Rodrigues, J.J., Righi, R.D.R., Kozlov, S. and Rodrigues, V.F., (2020). Looking at Fog Computing for E-Health through the Lens of Deployment Challenges and Applications. Sensors, 20(9), p.2553.

Wikipedia. (2019) Medical state - wikipedia. https://en.wikipedia.org/wiki/Medical_state,. (Accessed on 10/01/2019).

Zaima, H., Tsumori, O., Ono, S. and Ueda, T., Sharp Corp, (2009). Data converter, data conversion method, program for making computer function as data converter and recording medium for storing this program. U.S. Patent 7,535,477.

Zhang, W., Yang, J., Su, H., Kumar, M. and Mao, Y., (2018). Medical data fusion algorithm based on internet of things. Personal and Ubiquitous Computing, 22(5-6), pp.895-902.