A Fault Tolerant Data Management Scheme for Healthcare Internet of Things in Fog Computing

Waqar Saeed[1], Zulfiqar Ahmad[1*], Ali I. Jehangiri[1], Nader Mohamed[2*], Arif I. Umar[1], and Jamil Ahmad[1]

[1] Department of Information Technology
Hazara University, Mansehra, KPK, Pakistan
[e-mail: waqar.it@hu.edu.pk, zulfiqarahmad526@gmail.com, ali_imran@hu.edu.pk, arifiqbalumar@yahoo.com, jamil@ieee.org]
[2] Department of Computer Science, Information Systems, and Engineering
California University of Pennsylvania, California, PA 15419, USA
[e-mail: nader@middleware-tech.net]
*Corresponding authors: Zulfiqar Ahmad and Nader Mohamed

Received July 12, 2020; revised October 9, 2020; accepted December 13, 2020; published January 31, 2021

Abstract

Fog computing aims to provide the solution of bandwidth, network latency and energy consumption problems of cloud computing. Likewise, management of data generated by healthcare IoT devices is one of the significant applications of fog computing. Huge amount of data is being generated by healthcare IoT devices and such types of data is required to be managed efficiently, with low latency, without failure, and with minimum energy consumption and low cost. Failures of task or node can cause more latency, maximum energy consumption and high cost. Thus, a failure free, cost efficient, and energy aware management and scheduling scheme for data generated by healthcare IoT devices not only improves the performance of the system but also saves the precious lives of patients because of due to minimum latency and provision of fault tolerance. Therefore, to address all such challenges with regard to data management and fault tolerance, we have presented a Fault Tolerant Data management (FTDM) scheme for healthcare IoT in fog computing. In FTDM, the data generated by healthcare IoT devices is efficiently organized and managed through well-defined components and steps. A two way fault-tolerant mechanism i.e., task-based fault-tolerance and node-based fault-tolerance, is provided in FTDM through which failure of tasks and nodes are managed. The paper considers energy consumption, execution cost, network usage, latency, and execution time as performance evaluation parameters. The simulation results show significantly improvements which are performed using iFogSim. Further, the simulation results show that the proposed FTDM strategy reduces energy consumption 3.97%, execution cost 5.09%, network usage 25.88%, latency 44.15% and execution time 48.89% as compared with existing Greedy Knapsack Scheduling (GKS) strategy. Moreover, it is worthwhile to mention that sometimes the patients are required to be treated remotely due to non-availability of facilities or due to some infectious diseases such as COVID-19. Thus, in such circumstances, the proposed strategy is significantly efficient.

Keywords: Internet of Things, Healthcare, COVID-19, Fog Computing, Fault Tolerance, Data Management

http://doi.org/10.3837/tiis.2021.01.003 **ISSN : 1976-7277**

1. Introduction

Internet of Things (IoT) is an emerging environment in which all the surrounding smart devices/things are connected and communicate to each other via the internet. Most of the physical devices such as; intelligent ovens, smart healthcare devices, smart home appliances, smart watch, smart kegs, and drones are connected through the internet. These IoT devices produce huge amounts of data, which is being generated in each and every instance of seconds [1]. By 2020, CISCO expects 50 billion connected IoT devices with almost 7 IoT devices per person [2]. One of the prominent uses of IoT devices is in healthcare because IoT devices are useful to manage both patients and diseases. Better management is also required because of limited number of medical staff and facilities [3]. However, with the connection of devices at such a large scales, the overhead cost of communication will increase. The traditional clouds, due to increase of workload, are suffering from bandwidth, energy consumption, robustness, delay and latency challenges [4-6]. In order to overcome these challenges, a new emerging paradigm 'fog computing' has been introduced which aims to improve bandwidth, delay, and latency problems of cloud effectively and efficiently [7-10]. However, fog computing being a new computing paradigm to organize connections between devices also suffers from efficiency issues because of large number of connections. Therefore, this research proposes systematically approach to make fog computing more efficient and tolerant in respect of data management for healthcare IoT in fog computing and provision of two level fault tolerance at tasks and devices/nodes level. A novel scheme for Fault Tolerant Data management (FTDM) for healthcare IoT in Fog Computing is proposed in this paper. In addition to this, the paper also provides task-retry fault tolerant mechanism in FTDM at tasks level and tasks-transfer fault tolerant mechanism at fog devices/nodes level. The energy consumption, execution cost, network usage, latency (network delay), and execution time (make-span) as performance evaluation parameters are also considered in the research work. To evaluate the effectiveness of FTDM, we carried out a simulation using iFogSim [11] and then we compared the results of FTDM with existing strategy GKS [12].

Fog computing is a highly virtualized platform that provides computation, storage, and network services between IoT devices and traditional cloud computing datacenters [13]. In the case of healthcare IoT, fog computing still requires to reduce latency, energy consumption, and to provide efficient fault tolerant mechanisms [12, 14, 15]. So, all these limitations attract us to show some contribution in respect of efficient data management for healthcare IoT in fog computing.

When data is required to organize and manage efficiently and tasks are required to execute effectively and robustly for healthcare IoT in fog computing, many challenges need to be addressed [12, 15], such as:

- Suppose healthcare IoT devices generated "n" number of data packets i.e., D_{P1}, D_{P2}, D_{P3},…, D_{Pn} and required to be collected, organized and managed efficiently at "n" number of fog devices i.e., F_{D1}, F_{D2}, F_{D3}, …, F_{Dn} for decision/action. Therefore, a data management scheme is essential for healthcare IoT data in fog computing.
- The data collected at "n" number of fog devices i.e., F_{D1}, F_{D2}, F_{D3}, … F_{Dn} contains tasks i.e., T_1, T_2, T_3, …, T_n and these tasks are required to be executed effectively and robustly. Therefore, a fault tolerant mechanism is essential at tasks level and at nodes level.

In our work we systematically inquire and solve the above mentioned challenges in respect of data management for healthcare IoT in fog computing and provision of two level fault tolerance at tasks and devices/nodes level. Therefore, we proposed a Fault Tolerant Data management (FTDM) scheme for healthcare IoT in Fog Computing. The motivation of our work is to answer the basic question of concern to researchers: How much performance will be improved, if we provide a fault tolerant data management scheme for Healthcare IoT in fog computing?

Rest of the paper is organized as follows: Section 2 presents the related work. The need and importance of healthcare in IoT is provided in Section 3. Section 4 provides the system design and model. Section 5 presents evaluation methods including simulation tool, application modelling and performance evaluation parameters. Section 6 presents experimental setup, results and discussion. Finally, section 7 concludes the paper.

2. Related Work

We have examined the literature review in respect of data management and fault tolerance for IoT based healthcare devices in fog computing.

This section present the literature review which is divided into two parts, in the first part, a brief literature on data management, task management, and energy and latency-aware scheduling for healthcare IoT in fog computing is presented. Whereas, in the second part, various fault tolerant scheduling techniques in fog computing are discussed.

A low-latency and energy-efficient scheduling scheme Greedy Knapsack Scheduling (GKS) was proposed in [12]. GKS scheme is designed for fog-based IoT applications and it is used to reduce the latency and minimize energy consumption. Although the GKS scheme reduces latency and minimizes energy consumption but the authors are unable to provide the solution for failure of nodes and tasks in the proposed scheme. Similarly, the GKS Scheme is a generic scheme and it is not specifically designed for any particular area such as healthcare IoT.

In [15], an intelligence scheme is proposed for fog computing to achieve low energy consumption and latency reduction. The authors used an optimization algorithm for end users to provide the decision of tasks offloading in presence of multiple fog nodes. Although the proposed scheme solves the problem of latency and energy consumption, however, the problems of data management, tasks management, and fault tolerance in the proposed scheme remains intact.

In [16], a technique was proposed in order to convert the data into chunks to reduce the latency and improve the security for fog computing. The aim of converting data to chunks is to protect the whole data form malicious users and save the network bandwidth. Since, in an IoT environment, the reliability of data and fault tolerant parameters are very important because they survive the failures at edge servers and the said parameters have not been specifically considered in the proposed technique. In [17], an IoT edge architecture was proposed in which the cloud is distributed into four levels such as; cloud, fog, mist, and dew. The architecture was based on the processing power and distance from the end of IoT devices. The proposed architecture also makes the whole system reliable because of repeating the data at the edge of the network for IoT devices.

In [18], a scheme was presented for health care IoT to reduce latency and save bandwidth by combining fog computing and edge computing. But in this scheme, there are still some issues like memory and data management, QoS, and security. In [19], a new architecture was proposed to store big data and process the sensor data by using Meta Fog Redirection (MF-R), Grouping and Choosing (GC) architectures. In MF-R authors used Apache pig and apache

HBase for collection and storage of sensor data. GC architecture is used to secure the fog computing and cloud computing categorized data. The architecture of the proposed scheme consists of three phases, first one is data collection phase, in which the data is collected from the patient and sent alert to the doctor through wireless. Second phase is the data transfer phase in which clinical data is transferred to Amazon. The third phase is the Big data phase, if amazon cannot store data, the big data phase uses the Apache pig for storing the clinical data. Scheme have some limitations like, if the main database is temporarily down or crashed then there is no backup for data recovery.

In [20, 21], the mechanisms were proposed for healthcare monitoring by using the IoT concept to connect the patient with the internet. The limitations of the schemes are bandwidth and there is no fault tolerance. In [22], a fault tolerant technique in fog computing for healthcare IoT devices was presented to improve network reliability and processing speed. The authors in the proposed technique used Random Variable Neighboring Search (RVNS) based Computing and Analyzing (RCA) schemes that randomly choose the patient data. In this article, the authors used the fault tolerant technique to attain the connectivity of the nodes when they are failed due to high receiving data rate or other problem. The mechanism used for collecting the data of patients in the proposed scheme is not efficient in case of healthcare because of its critical nature. In [23], a fault tolerant strategy is proposed, in which the authors used the sink node as a fault tolerant node for monitoring the data in real time. There is a large number of data present in a single Gateway for healthcare monitoring systems, thus, it is difficult for a single sink node to monitor a large data at bottleneck.

In [24], it was argued that in cloud computing failures occur when they keep away from the fault tolerant mechanisms for the purpose of maintaining the financial profit. In this paper the author provided the HEFT (Heterogeneous Earliest Finish Time) scheme through which they used the proactive and reactive techniques to provide the hybrid fault tolerant strategy for cloud computing systems. In proactive strategy, the failure probability and usage time of the Virtual Machine (VM) is considered. In the reactive technique, task-replication and check-pointing strategies are applied. In general, proactive, reactive and resubmission mechanisms are used for fault tolerance in cloud platforms [25]. The proactive technique requires more information about the cloud computing and works in a probabilistic manner. In cloud computing the proactive strategy reduces the failure time and also increases the capacity and throughput. Whereas, reactive fault tolerance reduces the impact of failure on application execution when the failure efficaciously occurs. For scientific workflow systems, a study on fault tolerant mechanism [26] was proposed i.e., Cloud Outage Study (COS) in which it was argued that for scientific workflows mostly the resubmission mechanism is used as fault tolerance technique. Whenever a failed task is observed, it is resubmitted either to the same or to a separate resource at runtime.

In [27], a virtualization and fault-tolerance technique (VFT) was presented that is used to improve the system availability and also reduce the service time. In this scheme, the authors used the two modules i.e., Decision Maker and Cloud Manager, to manage the load balancing, virtualization and also to handle the faults. In this paper, the fault tolerance is used for real time cloud computing. When the system is running to the fault, the decision is taken on the basis of reliability of the processing nodes known as virtual machines. The reliability of the virtual machines is changing after every cycle, if the virtual machine manages to provide a rectify result within the stipulated time, then the reliability of the system will be increased. If it is failed to produce the result within stipulated time, the reliability of the system will be decreased. In this scheme, the authors aim to increase the level of reliability at very low cost. If any of the nodes is failed, it is removed, and a new node will be added, therefore, for such situations there will be a minimum level of reliability which was explained in the technique Adaptive Fault Tolerance in Real-time Cloud Computing (AFTRC) [28]. Since, the large amount of data is stored in cloud computing, thus, in [29], a dynamic data fault tolerance mechanism was proposed for cloud storage i.e., Dynamically Determine Fault-tolerant Mechanism Conservation System (DDFMCS). The proposed mechanism is implemented in Hadoop by conducting some experiments. The experimental result shows that the DDFMCS improves the data access performance and minimizes the utilization of cloud storage space.

3. Healthcare in IoT

Healthcare is an indispensable part of the human body for life [30, 31]. But, unfortunately due to the development of population, senescent persons facing many challenges in respect of healthcare services [32]. For example, the rehabilitation services after accidents and services for the over-aged persons [33]. Since, due to the increasing number of patients, there is also required high number of beds in a hospital and more number of doctors are required to check each and every patient [34, 35]. It is also worthwhile to mention that sometimes the patients are required to be treated remotely due to non-availability of facilities or due to an infectious disease such as COVID-19 [36]. Eventually, a solution is required to: manage the patients' data efficiently, provide high quality of service (QoS), and provision of treatments' facilities to a maximum number of patients within a minimum time. Therefore; an automatic, independent and fault-tolerant system is necessary, as such we have presented FTDM by using the internet of things (IoT) in the healthcare system [37].

There are several numbers of IoT devices/nodes which are configured/installed at various body parts of a person/patient for collection of data. The persons/patients are present at multiple locations with heterogeneous environmental conditions, therefore, there are maximum chances of failure of nodes/IoT devices. Secondly, a huge amount of data for different categories of healthcare domain is collected through IoT devices/nodes which are heterogeneous in nature and are executed/evaluated at cloud resources. As such there are also chances of failure of tasks in fog environments. All these deficiencies are not considered by the authors in the existing GKS [12] scheme i.e., there is no efficient data management and fault tolerant mechanism for healthcare tasks and IoT devices. If some tasks are failed or some nodes are failed and unable to send data timely and efficiently, the energy consumption and latency will be maximized. Thus, a failure free, cost efficient, and energy aware management and scheduling scheme for data generated by healthcare IoT devices not only improves the performance of the system but, due to minimum latency and provision of fault tolerance, also saves the precious lives of patients. Therefore, to address all such challenges with regard to data management and fault tolerance, a Fault Tolerant Data management (FTDM) scheme for healthcare IoT in fog computing is presented. The presented FTDM scheme will be an efficient solution for fault tolerant aware data management and to reduce energy consumption and latency for healthcare IoT in fog computing.

4. System Design and Model

In this section the design of the proposed model is presented. Fig. 1 shows the overall architecture of FTDM. There are four core components of FTDM, i.e., (a) Data Collector, (b) Data Analyzer and Scheduler, (c) Fault Tolerant Provider, and (d) Decision Maker. The FTDM scheme not only organizes and manages healthcare data (such as Blood Pressure, RespiratoryRate, Heart Rate and Body Temperature) generated by IoT devices but also provides two level fault tolerance at tasks and devices/nodes level.

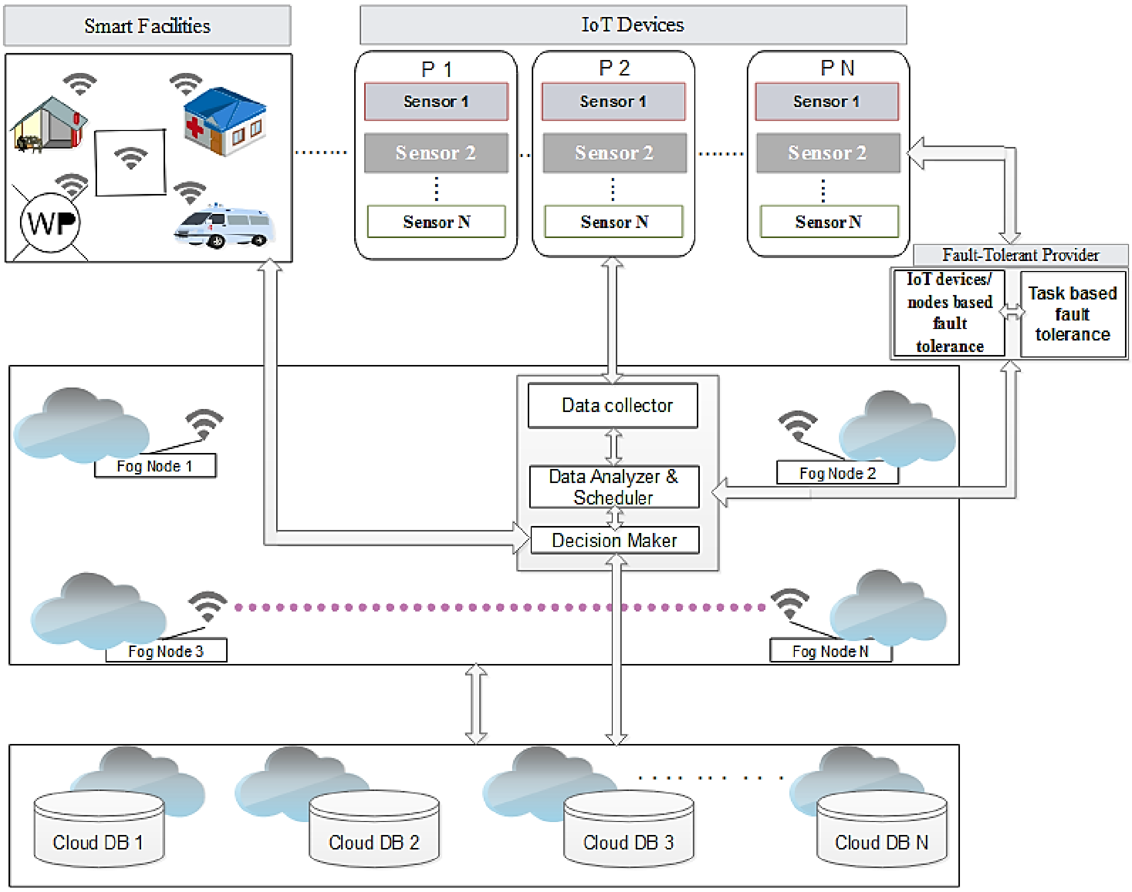


Fig. 1. Fault-tolerant Data Management Scheme

The “Data Collector” component is responsible for collection of healthcare data generated by the IoT devices related to the patients. Then the “Data Analyzer and Scheduler” component examines and schedules the collected data. This component is also responsible for allocation of tasks from data to the appropriate resources in fog computing environments. If the tasks are successfully executed, the result will be sent to the “Decision Maker” component, otherwise, Fault Tolerant Provider component will be executed. The “Fault Tolerant Provider” component performs two essential fault tolerant mechanisms at this level i.e., “Task-based fault tolerance” and “IoT devices/nodes based fault tolerance”. Firstly, in “Task-based fault tolerance”, if the tasks are failed at any node, it will be re-executed at other node, since, it is assumed that the Scheduler will retain a copy of the task when it is assigned to the fog nodes until its complete execution. Secondly, in “IoT devices/nodes based fault tolerance”, if any of the nodes in a fog environment is failed, then the tasks on the failed node will be transferred to another working node by utilizing the retained copy of tasks. The “Decision Maker” component receives the analyzed/executed data from “Data Analyzer”. This component classifies the data into two parts, i.e., normal and abnormal data. The normal data is the data that is appropriate for patient health and will then be stored in a datacenter. The abnormal data is the inappropriate data which is unsafe for patient health. When abnormal data is received by “Decision Maker”, it will generate an alarm for action to the patient device as well as on the device of the patient’s consultant.

We have considered the fog computing environment in which there are various arrangements of sensor nodes, edge devices, and cloud datacenters. The sensors are located at the end of a geographic location for the end user (patients in case of healthcare environment) to collect the healthcare data. For the application scenario we assumed that some sensors are deployed in the human body (for healthcare) that monitor the condition of patients and also collect healthcare data of the patients. Initially the healthcare data will be stored in a fog database for decision making which is then permanently stored on cloud datacenters for future perspectives. The step-by-step mechanism of FTDM scheme is described in Algorithm 1.

Algorithm 1 Fault-Tolerant Data Managment (FTDM)

```
Input: $\alpha$ (Healthcare IoT Data)
Output: $\beta$ (Organized/Decision Based Healthcare IoT data)
procedure FTDM($\alpha$)
    Data Collector ()                                   ▷ Healthcare IoT Data Collection
    while (Data Analyzer and Scheduler () if(task/node failed)) then
do                                                      ▷ Analysis of IoT Data
FaultTolerantProvider()
        Decision Making ()
        Data storage for $\beta$
    Exit
```

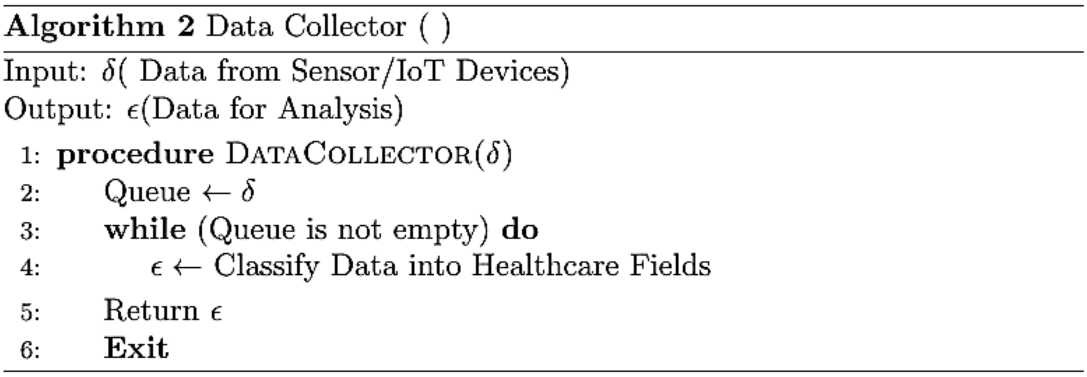
Algorithm 1 provides the step-by-step mechanism of FTDM. Initially, healthcare IoT data such as; respiratory rate, heart rate, blood pressure, body temperature and blood sugar [19], is collected through the Data Collector component by using IoT devices. The Data Collector after collection of raw data will then classify the same into required fields such as; respiratory rate, heart rate, blood pressure, body temperature and blood sugar. The Data Analyzer and Scheduler component will be then applied on the healthcare IoT data in order to examine and schedule it. The data is examined in a sense that various types of data such as respiratory rate, heart rate, blood pressure, body temperature and blood sugar [19] is being collected from the patient body through IoT devices. Different processes are being designed to analyze such types of data as to whether the healthcare data lies within the normal human range or not. Such type of process are required to be executed on computing resources. At this stage, examined data which is now converted into various tasks are scheduled to the required resources for execution. If the tasks are failed to execute or the node to which the tasks are assigned is failed, the Fault Tolerant Provide mechanism will be executed. For tasks, the Fault Tolerant Provide will simply re-execute them. On the other hand, since, there are a number of fog nodes available for tasks allocations. Therefore, if Scheduler assigns tasks to a particular fog node and if that fog node fails, the Scheduler will execute another copy of that task through another node. In the case of proposed work, it is assumed that the Scheduler will retain a copy of the task when it is assigned to the fog nodes until its complete execution. After successful examination of data and execution of tasks, the data is then sent to Decision Maker. In this stage, the data is classified into two parts, i.e., normal and abnormal data. The normal data is the appropriate data which is suitable for patient health, this data will be then stored to the cloud datacenters by executing the Data Storage module. The abnormal data is the inappropriate data which is unsafe for patient health. When abnormal data is received by “Decision Maker”, it will generate an alarm for action to the patient device as well as on the device of the patient’s consultant.

4.1 Components of FTDM

The FTDM scheme is classified into the four core components. These components are: (a) Data Collector, (b) Data Analyzer and Scheduler, (c) Fault Tolerant Provider, and (d) Decision Maker. The detailed discussion along with appropriate algorithm for each component is given below.

4.1.1 Data Collector

Data Collector component collect the healthcare data from IoT devices and classified it into the proper format as per healthcare fields. The step-by-step procedure of Data Collector component is shown in Algorithm 2.



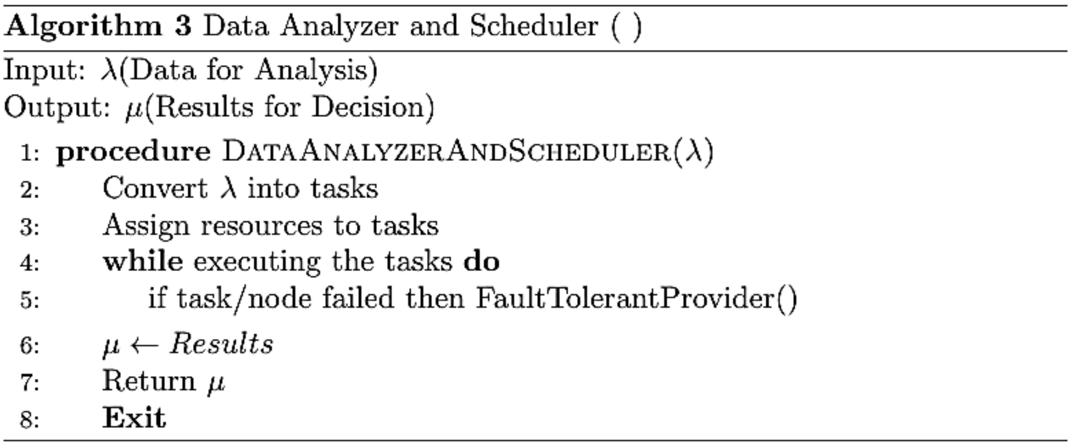
Algorithm 2 Data Collector ( )

```
Input: $\delta$( Data from Sensor/IoT Devices)
Output: $\epsilon$(Data for Analysis)
procedure DATACOLLECTOR($\delta$)
    Queue $\leftarrow \delta$
    while (Queue is not empty) do
        $\epsilon \leftarrow$ Classify Data into Healthcare Fields
    Return $\epsilon$
    Exit
```

Algorithm 2 provides the step-by-step procedure of Data Collector component. The healthcare IoT data generated by IoT devices such as; respiratory rate, heart rate, blood pressure, body temperature and blood sugar [19], is collected through Data Collector component by using IoT devices. The Data Collector after collection of raw data will then classify the same into required fields such as; respiratory rate, heart rate, blood pressure, body temperature and blood sugar. The classified data is then will be used by the Data Analyzer and Scheduler component for examining and scheduling the same.

4.1.2 Data Analyzer and Scheduler

Data Analyzer and Scheduler after obtaining the healthcare IoT data from Data Collector in the form of healthcare IoT tasks will examine and schedule them to the required resources. If all the tasks in the data are successfully executed, the data will be then sent to the Decision Maker, otherwise, the module of Fault Tolerant Provider will be executed. The step-by-step procedure of Data Analyzer and Scheduler component is shown in Algorithm 3.



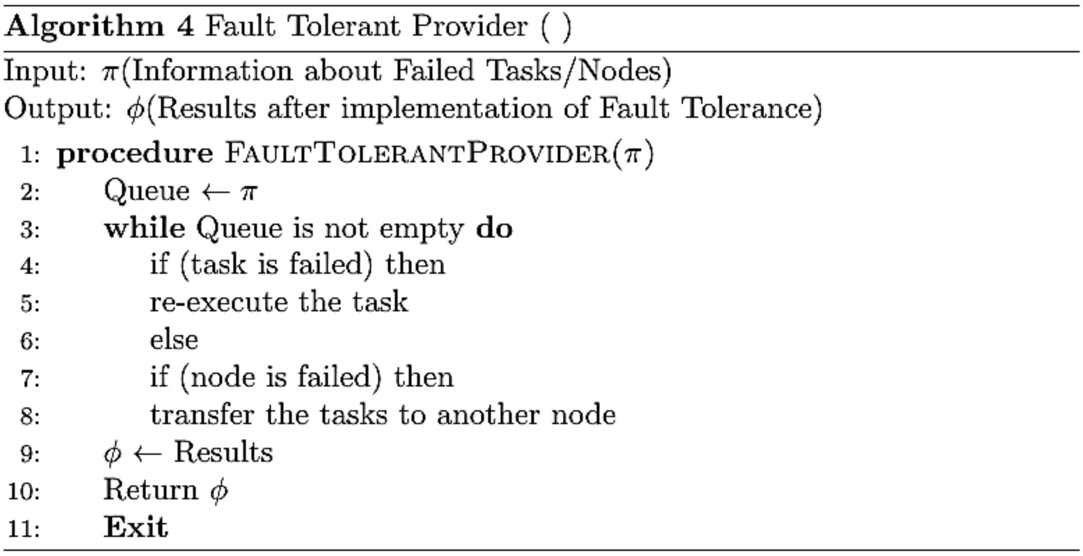
Algorithm 3 Data Analyzer and Scheduler ( )

```
Input: $\lambda$(Data for Analysis)
Output: $\mu$(Results for Decision)
procedure DataAnalyzerAndScheduler($\lambda$)
    Convert $\lambda$ into tasks
    Assign resources to tasks
    while executing the tasks do
        if task/node failed then FaultTolerantProvider()
    $\mu \leftarrow Results$
    Return $\mu$
    Exit
```

Algorithm 3 provides the step-by-step procedure of Data Analyzer and Scheduler component. Data Analyzer and Scheduler after obtaining the classified healthcare IoT data in the form of healthcare IoT tasks from Data Collector will examine and schedule them to the required resources. The data is examined in a sense that various types of data such as respiratory rate, heart rate, blood pressure, body temperature and blood sugar [19] is being collected from the patient body through IoT devices. Different processes are being designed to analyze such types of data as to whether the healthcare data is lies within the normal human range or not. Such types of processes are required to be executed on computing resources. At this stage, examined data which is now converted into various tasks are scheduled to the required resources for execution. If the tasks are failed to execute or the node to which the tasks are assigned is failed, the Fault Tolerant Provide mechanism will be executed. For tasks, the Fault Tolerant Provide will simply re-execute them. On the other hand, since, there are a number of fog nodes available for tasks allocations. Therefore, if Scheduler assigns tasks to a particular fog node and if that fog node fails, the Scheduler will execute another copy of that task through another node. In the case of proposed work, it is assumed that the Scheduler will retain a copy of task when it is assigned to the fog nodes until its complete execution. After successful examination of data and execution of tasks, the data is then sent to Decision Maker.

4.1.3 Fault Tolerant Provider

If the tasks (that are assigned to the required resources) are failed to execute or the node to which the tasks are assigned is failed, the Fault Tolerant Provider mechanism will be executed. The “Fault Tolerant Provider” component perform two essential fault tolerant mechanisms i.e., “Task-based fault tolerance” and “IoT devices/nodes based fault tolerance”. Firstly, in “Task-based fault tolerance”, if the tasks are failed at any node, it will re-execute the failed tasks. Secondly, in “Task-based fault tolerance”, if any of the node in fog environment will be failed, then the tasks on the failed node will be transferred to other working node for execution.



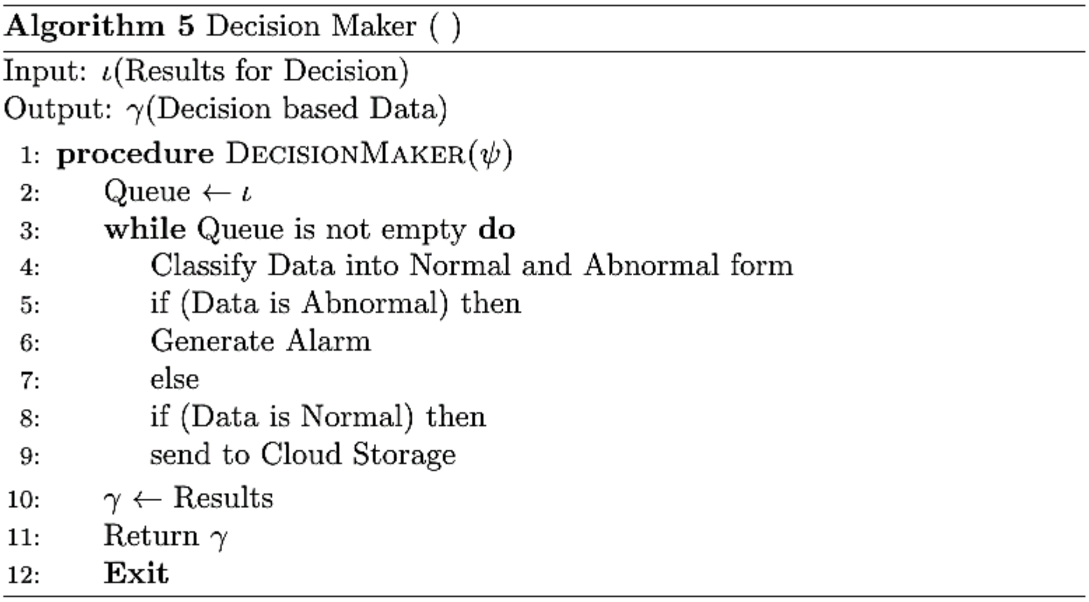
Algorithm 4 Fault Tolerant Provider ( )

```
Input: $\pi$(Information about Failed Tasks/Nodes)
Output: $\phi$(Results after implementation of Fault Tolerance)
procedure FaultTolerantProvider($\pi$)
    Queue $\leftarrow \pi$
    while Queue is not empty do
        if (task is failed) then
        re-execute the task
        else
        if (node is failed) then
        transfer the tasks to another node
    $\phi \leftarrow$ Results
    Return $\phi$
    Exit
```

Algorithm 4 provides the step-by-step procedure of Fault Tolerant Provider component. There are two essential components of "Fault Tolerant Provider" i.e., "Task-based fault tolerance" and "IoT devices/nodes based fault tolerance". In "Task-based fault tolerance", if the tasks are failed at any node, it will re-execute the failed tasks. On the other hand, since, there are a number of fog nodes available for tasks allocations. Therefore, if Scheduler assigns tasks to a particular fog node and if that fog node fails, the Scheduler will execute another copy of that task through another node. As in the case of proposed work, it is assumed that the Scheduler will retain a copy of the task when it is assigned to the fog nodes until its complete execution. So far as the determination of failure of task/node is concerned, the proposed model is working on healthcare IoT data in which the response time from IoT devices and fog nodes is one of the important component. Therefore, if the IoT devices or fog nodes will not response within the short range of stipulated time, it will be assumed that there is problem at that very IoT device or fog nodes. As such, the Fault Tolerant Provider component will be invoked.

4.1.4 Decision Maker

After successful examination of data and execution of tasks, the data is received by Decision Maker. In decision making, the data is classified into two parts, i.e., normal and abnormal data. The normal data is suitable for patient health, this data will be then sent and stored to the cloud datacenters. The abnormal data is unsafe data for patient health. The abnormal data includes abnormal heart beat/rate, respiration rate, and body temperature and when this type of data is received by "Decision Maker", it will generate an alarm for action to the patient device as well as on the device of the patient's consultant.



Algorithm 5 Decision Maker ( )

```
Input: $\iota$(Results for Decision)
Output: $\gamma$(Decision based Data)
procedure DecisionMaker($\psi$)
    Queue $\leftarrow \iota$
    while Queue is not empty do
        Classify Data into Normal and Abnormal form
        if (Data is Abnormal) then
        Generate Alarm
        else
        if (Data is Normal) then
        send to Cloud Storage
    $\gamma \leftarrow$ Results
    Return $\gamma$
    Exit
```

Algorithm 5 provides the step-by-step procedure of Decision Maker component. In this stage, the data is classified into two parts, i.e., normal and abnormal data. The normal data is the appropriate data which is suitable for patient health, this data will be then stored to the cloud datacenters by executing the Data Storage module. The abnormal data is the inappropriate data which is unsafe for patient health. When abnormal data is received by "Decision Maker", it will generate an alarm for action to the patient device as well as on the device of the patient's consultant.

4.2 Dataflow in FTDM

The flow of data in FTDM is shown in **Fig. 2**. The data will be collected by Data Collector from the patients through IoT devices and will be initially stored at fog devices/database. The collected data then sent to Data Analyzer and Scheduler, where the data will be examined for correctness and tasks are assigned to the resources for execution.

If any of the tasks or nodes are failed, the Fault Tolerant Provider component, re-execute the failed tasks or transfer the tasks on failed node to some other nodes. After successful examination and execution of data, the data will be sent to the Decision Maker for appropriate decision. At this stage, the data will be classified into two parts, normal and abnormal data. Normal data is safe data for patients' health and will be stored at cloud datacenters for future use, whereas, when abnormal data is identified, an alarm will be generated at patient's IoT device as well as on IoT device of the patient's consultant.

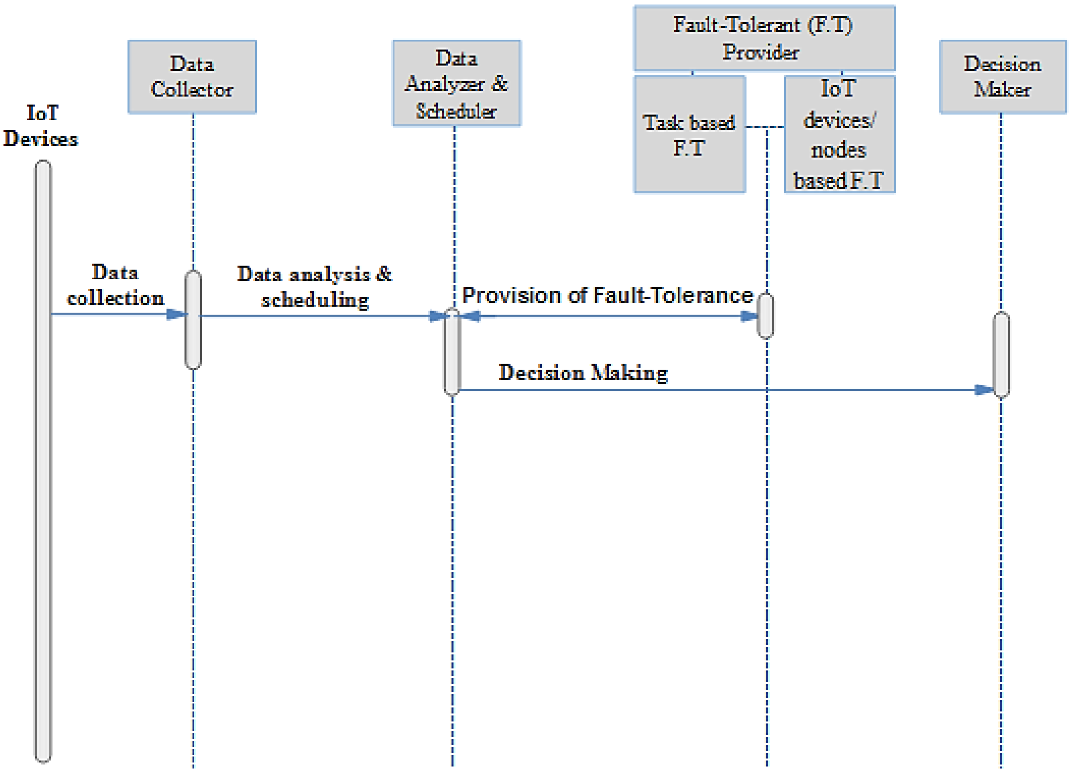

IoT Devices
Data Collector
Data Analyzer & Scheduler
Fault-Tolerant (F.T) Provider
Task based F.T
IoT devices/ nodes based F.T
Decision Maker
Data collection
Data analysis & scheduling
Provision of Fault-Tolerance
Decision Making

Fig. 2. Sequence Model of FTDM

Since, fog computing is a highly virtualized computing platform that provides resources in the form of a large number of geographically available fog nodes [13]. Various processing and computational tasks are being processed on fog computing nodes. In cloud and fog environments, failures of nodes and tasks occur when they keep away from the fault tolerant mechanisms for the purpose of maintaining the financial profit [24]. As such we provided a two level fault tolerant mechanism i.e., “Task-based fault tolerance” and “IoT devices/nodes based fault tolerance”. Firstly, in “Task-based fault tolerance”, if the tasks are failed at any node, it will be re-executed at other node, since, it is assumed that the Scheduler will retain a copy of the task when it is assigned to the fog nodes until its complete execution. Secondly, in “IoT devices/nodes based fault tolerance”, if any of the node in fog environment is failed, then the tasks on the failed node will be re-executed from other working nodes by utilizing the retain copy of tasks. The proposed work is simulation based work, in which the fog computing simulator i.e., iFogSim [11] is used to evaluate the presented strategy FTDM.

5. Evaluation Methods

In this section, a brief description on simulation tool, application modeling, and performance evaluation parameters, is presented.

5.1 Simulation Tool

To enable modeling and simulation of fog computing environment for our proposed scheme, we have used iFogSim [11]. The iFogSim is a simulation tool that combine the cloud datacenters with fog environment and IoT devices to establish IoT, fog and cloud computing scenarios.

5.2 Application Modelling

In order to show the efficiency of the proposed FTDM scheme, we have executed the VR (Virtual Reality) Game, as executed by the existing GKS [19] scheme. VR Game is the application of a three dimensional artificial environment to computer games. It is created with VR software and presented to the user in such a way that it provides the real world environment. The VR Game is also important in healthcare because stroke rehabilitation is one of the most successful health research area where virtual reality technologies and therapeutic serious games have been used to create new intervention tools. This game is not only used for fun but also plays a vital role in physical exercise in order to achieve healthcare goals [38, 39].

5.3 Performance Evaluation Parameters

We have used energy consumption, execution cost, network usage, network delay, and execution time as performance evaluation parameters. Minimum latency/network delay and less execution time are the prime goals of the healthcare IoT environment. Similarly, less energy consumption, minimum execution cost, and less network usage are the current requirements of fog computing in respect of healthcare service.

5.3.1 Energy Consumption

Energy consumption is an important aspect for IoT devices because many sensor devices are bounded by battery durations. When a task is executed by taking a specific time for the processing of data, energy is consumed and its measurement unit is joule (J). The sensor devices consumed the maximum energy during the transmission, sensing and execution of tasks [40]. The energy consumption of the proposed system has been calculated with the help of (1).

$$Energy_{cons} = \sum_{i=1}^{n} E_{trans(i)} + E_{exe(i)} + E_{sen(i)} \tag{1}$$

The $Energy_{cons}$ is the total energy consumed, whereas, E_{trans} is energy consumed during transmission of each individual task, E_{exe} is the energy consumption on execution of each individual task, and E_{sen} is the energy consumption on sensing of each individual task.

5.3.2 Execution Cost

Execution cost is the cost of total MIPS (million instructions per second) of hosts with respect to the time frame [12, 41]. The execution cost is measured in dollars and it has been calculated with the help of (2).

$$Execution_{cost} = \sum_{i=1}^{n} Host\ (i)\ _{MIPS} \times TFrame\ _{HOST\ (i)} \times Cost\ _{HOST\ (i)} \tag{2}$$

Execution$_{\text{cost}}$ represents the total Execution cost, whereas, Host$_{\text{MIPS}}$ represents the total number of MIPS on each individual host, TFrame$_{\text{HOST}}$ represents the time frame taken by each individual host and Cost$_{\text{HOST}}$ represents the cost of each individual host.

5.3.3 Network Usage

In fog computing, when the number of devices connected to the application are increased, the load on the network will also increase [42]. The network usage is calculated in kilobytes and it is load on a network. The network usage is calculated by using (3).

$$Network_{usage} = \sum_{i=1}^{n} Network_{Load(Ni)} \times No.of\ Messages(Ni) \tag{3}$$

The *Network*$_{\text{usage}}$ represents the network usage, whereas, Network$_{\text{Load}}$ represents network load on each individual node and No. of Messages (N) represents the total number of messages for each individual node.

5.3.4 Network Delay

The network delay is calculated by adding all types of delay such as processing delay, transmission delay, and computation delay. The term latency is also used for network delay. The network delay is measured in millisecond (ms) [42] and it is calculated with the help of (4).

$$Network_{Delay} = P_D + T_D + C_D \tag{4}$$

The *Network*$_{\text{Delay}}$ represents the network delay, whereas, P$_{\text{D}}$ represents the processing delay, T$_{\text{D}}$ represents the transmission delay, and C$_{\text{D}}$ represents the computation delay.

5.3.5 Execution Time

Execution time is the total time taken by the task for processing and it is measured in seconds [41, 42]. The execution time is calculated by using (5).

$$Execution_{Time} = \sum_{i=1}^{n} \{Task(i)_{No.\ of\ Instructions} \times \sum_{j=1}^{n} Instructions(j)_{Time}\} \tag{5}$$

The $Execution_{Time}$ represents the execution time, whereas, $Task_{No.\ of\ Instruction}$ represents the total number of instruction for each individual task and $Instruction_{\mathrm{Time}}$ represents the time taken by each individual instruction of task.

6. Experimental Setup, Results and Discussion

This section represents the experimental setup followed by results and discussion, in which the results of performance evaluation parameters of the proposed system are discussed. The result and discussion section is further divided into five sub-sections i.e. energy consumption, cost of execution, network usage, network delay and execution time.

6.1 Experiment Setup

The simulation environment that we have used is iFogSim [43, 44]. The simulation was performed on a system with properties such as, CPU (Intel Core i5 2.50 Giga Hertz), RAM (8 Gigabyte), and OS (Microsoft Windows 10 64-bit). We have simulated the FTDM scheme and compared the result with existing scheduling technique GKS [12]. Since, the significance of proposed work is evaluated through simulation on the basis of existing and most relevant published work GKS [12], in which the similar iFogSim configurations were used. However, in the future we will test the proposed model with a real testbed and real datasets, while currently we are testing our algorithm using the existing algorithm datasets to prove the superior within the same datasets.

The comparisons were made in respect of performance evaluation parameters: energy consumption, execution cost, network usage, network delay, and execution time. We have simulated VRGame with six states of areas/departments, mobile/cameras, and Fog Devices (FDs) with {1 = (2, 6, 16), 2= (2, 7, 18), 3 = (3, 6, 23), 4 = (3, 7, 26), 5 = (4, 6, 30), 6 = (4, 7, 34)}, as the same configurations have been used by the existing technique GKS [12].

6.2 Results and Discussion

The simulation was performed for each performance evaluation parameter for up to 50 times and the average values were considered. The VRGame is simulated with six states of area/departments, mobile/cameras, and FDs with {1 = (2, 6, 16), 2= (2, 7, 18), 3 = (3, 6, 23), 4 = (3, 7, 26), 5 = (4, 6, 30), 6 = (4, 7, 34)}. The results for each performance evaluation parameter and discussion on results are given below:

6.2.1 Energy Consumption

The results in respect of energy consumption for proposed FTDM strategy are significant as compared with existing GKS strategy as reflected from **Fig. 3**. The proposed FTDM strategy reduces 3.97% energy consumption as compared with the existing GKS strategy. It is because of that, only the failed tasks are re-executed at the current stage in proposed FTDM strategy, whereas, in existing strategy when the tasks are failed, the execution will be started at initial stage of the system.

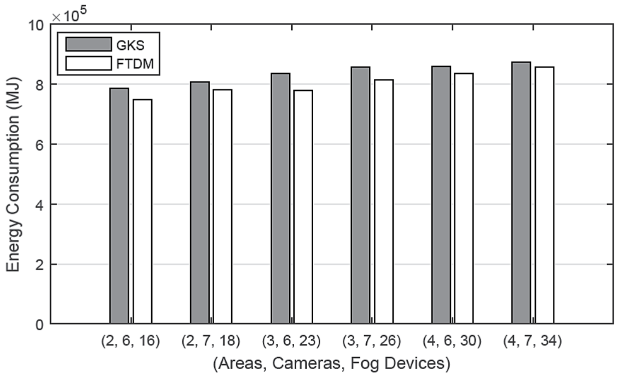


Fig. 3. Energy consumption comparison of GKS (existing) and FTDM (proposed) strategies

6.2.2 Cost of Execution

The results in respect of execution cost for proposed FTDM strategy are significant as compared with existing GKS strategy as reflected from **Fig. 4**. The proposed FTDM strategy reduces 5.09% execution cost as compared with the existing GKS strategy. It is because of that, the minimum number of resources are used in proposed FTDM strategy due to the provision of fault-tolerance.

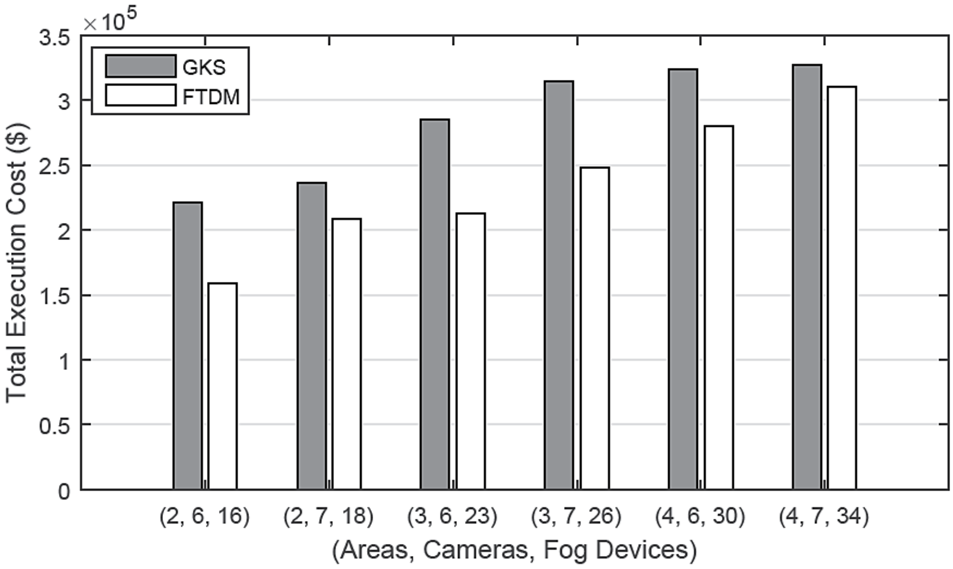


Fig. 4. Cost of Execution comparison of GKS (existing) and FTDM (proposed) strategies

6.2.3 Network Usage

The results in respect of network usage for proposed FTDM strategy are significant as compared with existing GKS strategy as reflected from **Fig. 5**. The proposed FTDM strategy reduces 25.88% network as compared with the existing GKS strategy. It is because of that, in proposed FTDM strategy less number of nodes are involved and as such there are less number of communications.

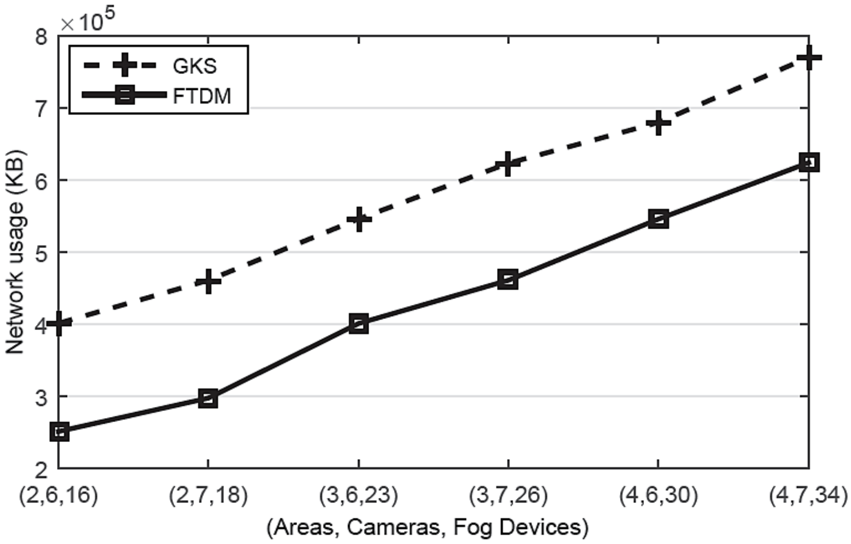


Fig. 5. Network Usage comparison of GKS (existing) and FTDM (proposed) strategies

6.2.4 Network Delay

The results in respect of network delay for proposed FTDM strategy are significant as compared with existing GKS strategy as reflected from **Fig. 6**. The proposed FTDM strategy reduces 44.15% network delay as compared with the existing GKS strategy. It is because of that, in proposed FTDM strategy, due the provision of fault tolerant mechanism, the failed tasks are re-executed immediately when they are failed, whereas, it is not the case in existing GKS strategy. Therefore, there will be minimum network delay for the proposed FTDM strategy.

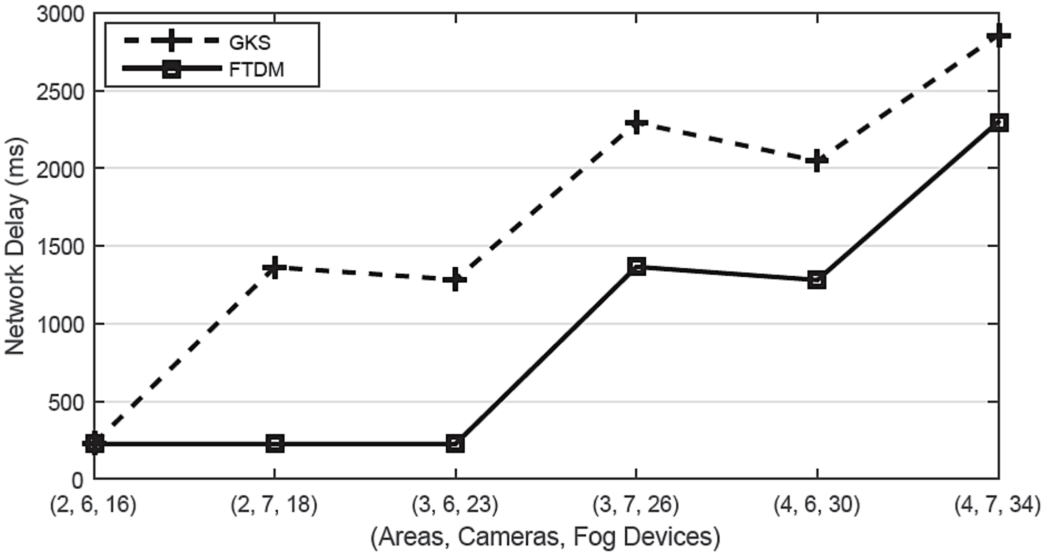


Fig. 6. Network Delay comparison of GKS (existing) and FTDM (proposed) strategies

6.2.5 Execution Time

The results in respect of execution time for proposed FTDM strategy are significant as compared with existing GKS strategy as reflected from **Fig. 7**. The proposed FTDM strategy reduces 48.89% execution time as compared with the existing GKS strategy. It is because of that, in the proposed FTDM strategy, due to less number of executions, minimum execution time has been taken for tasks.

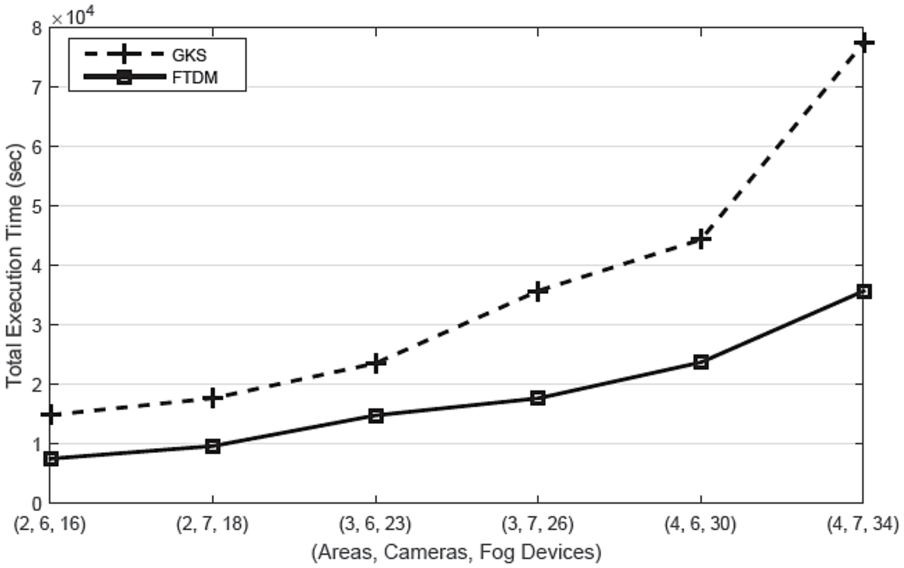


Fig. 7. Execution Time comparison of GKS (existing) and FTDM (proposed) strategies

The simulation results with regard to all the performance evaluation parameters for proposed FTDM strategy and existing GKS strategy is shown in **Table 1**. The results of the proposed FTDM strategy are compared with GKS strategy. It is because of that the GKS [12] scheme is designed for fog-based IoT applications and it is used to reduce the latency and minimize energy consumption. However, the GKS scheme is unable to provide the solution for failure of nodes and tasks. In cloud and fog environments, failures of nodes and tasks occur when they keep away from the fault tolerant mechanisms for the purpose of maintaining the financial profit [24]. As such, to overcome the discrepancies regarding failure we provide FTDM scheme based on the existing GKS scheme. The proposed FTDM scheme which is a fault tolerant based data management and scheduling scheme specifically designed for Healthcare IoT data. Therefore, in order to prove the superior within the same datasets and similar strategy, we compared the results of FTDM with GKS scheme.

Table 1. Simulation Results of VRGame Application

Areas	*Cameras*	*FD*	*GKS*	*FTDM*
			Energy consumption (MJ)	
2	6	16	25786523	25748581
2	7	18	25806766	25781069
3	6	23	25835025	25780350
3	7	26	25856206	25814858
4	6	30	25859674	25835749
4	7	34	25872638	25856903
			Total execution cost ($)	
2	6	16	4221428	4158896
2	7	18	4236551	4208406
3	6	23	4285456	4212696
3	7	26	4314113	4248013
4	6	30	4324032	4279885
4	7	34	4327108	4310295
			Total Network usage (KB)	
2	6	16	400947.2	250995.3
2	7	18	460639.7	297349
3	6	23	545646.2	400701
3	7	26	622907.9	460701.7
4	6	30	679427	545520.8
4	7	34	769580.6	623576
			Network Delay (ms)	
2	6	16	227.1968	226.1404
2	7	18	1363.466	226.7111
3	6	23	1284.039	227.1912
3	7	26	2294.101	1366.1
4	6	30	2046.877	1281.512
4	7	34	2858.536	2299.268
			Execution Time (Sec)	
2	6	16	14917.6	7567.5
2	7	18	17742.1	9698.6
3	6	23	23620.7	14859.9
3	7	26	35801.3	17727.2
4	6	30	44426.6	23781.1
4	7	34	77518.1	35762.2

The simulation results in respect of energy consumption, execution cost, network usage, network delay and execution time are shown in **Table 1**. The results reveal that almost for all the evaluation parameters, the proposed FTDM strategy is significantly better than the existing GKS strategy. The core reason behind the better performance of proposed FTDM is that in FTDM there are well-defined steps for efficient management of healthcare data with provision of suitable fault-tolerant mechanism. Therefore, the proposed FTDM strategy not only provides the mechanism of efficient data management generated by healthcare IoT devices but also provides an effective fault tolerant mechanism.

7. Conclusion

A systematically solution is provided in this paper to overcome the problems of data management, failure of tasks and nodes for healthcare IoT in fog computing by providing a novel scheme with two levels based fault tolerance i.e., (a) at tasks level, and (b) at devices/nodes level. The simulation results show that the newly developed model is more superior as compared to other similar model used for fog computing in health care systems. The major advantages of the proposed model are: reduce energy consumption 3.97%, execution cost 5.09%, network usage 25.88%, network delay 44.15%, and exaction time 48.89% as compared with the existing GKS strategy. Moreover, it is also worthwhile to mention that sometimes the patients are required to be treated remotely due to non-availability of facilities or due to some infectious diseases such as COVID-19. Thus, in such circumstances, the proposed strategy is significantly efficient.

This work will be extended by incorporating Service Level Agreement (SLA) based energy efficient data management technique for healthcare IoT in fog computing in the future.

Reference

[1] C. K. Dehury and P. K. Sahoo, “Design and implementation of a novel service management framework for IoT devices in cloud,” *Journal of Systems and Software*, vol. 119, pp. 149-161, 2016. Article (CrossRef Link)

[2] A. Brogi and S. Forti, “QoS-aware deployment of IoT applications through the fog,” *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185-1192, 2017. Article (CrossRef Link)

[3] F. A. Kraemer, A. E. Braten, N. Tamkittikhun, and D. Palma, “Fog computing in healthcare- A review and discussion,” *IEEE Access*, vol. 5, pp. 9206-9222, 2017. Article (CrossRef Link)

[4] P. Hu, “Survey on fog computing: architecture, key technologies, applications and open issues,” *Journal of network and computer applications*, vol. 98, pp. 27-42, 2017. Article (CrossRef Link)

[5] Z. Mahmood, “Data location and security issues in cloud computing,” in *Proc. of 2011 International Conference on Emerging Intelligent Data and Web Technologies*, pp. 49-54, 2011. Article (CrossRef Link)

[6] A. Mohammad, B. B. Rad, M. O. Thomas, and B. A. Onyimbo, “A Shift in Technological Paradigm: Cloud Computing to Fog computing,” *Journal of Engineering Science and Technology*, pp. 216-228, 2018. Article (CrossRef Link)

[7] K. Shabnam, S. Singh, and Radha, “Fog Computing: Characteristics and Challenges,” *International Journal of Emerging Trends & Technology in Computer Science*, vol. 6, no. 2, pp. 113-117, 2017. Article (CrossRef Link)

[8] Y. Jiang, Z. Huang, and D. H. K. Tsang, “Challenges and Solutions in Fog Computing Orchestration,” *IEEE Network*, vol. 32, no. 3, pp. 122-129, 2018. Article (CrossLink)

[9] A. A. Diro, N. Chilamkurti, and N. Kumar, “Lightweight Cybersecurity Schemes Using Elliptic Curve Cryptography in Publish-Subscribe Fog Computing,” *Mobile Networks and Applications*, vol. 22, no. 5, pp. 848-858, 2017. Article (CrossRef Link)

[10] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, “Towards fault tolerant fog computing for IoT-based smart city applications,” in *Proc. of 2019 IEEE 9th Annual Computing and Communication Workshop and Conference*, pp. 752-757, 2019. Article (CrossRef Link)

[11] 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,” *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275-1296, 2017. Article (CrossRef Link)

[12] D. Rahbari and N. Mohsen, “Low-latency and energy-efficient scheduling in fog-based IoT applications,” *Turkish Journal of Electrical Engineering and computer Sciences*, vol. 27, pp. 1406-1427, 2019. Article (CrossRef Link)

[13] C. V. N. Angeline and R. Lavanya, "Fog Computing and Its Role in the Internet of Things," *Advancing Consumer-Centric Fog Computing Architectures*, pp. 63-71, 2019. Article (CrossRef Link)

[14] O. Osanaiye, S. Chen, Z. Yan, R. Lu, K. Choo, and M. Dlodlo, "From cloud to fog computing: A review and a conceptual live VM migration framework," *IEEE Access*, vol. 5, pp. 8284-8300, 2017. Article (CrossRef Link)

[15] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," *Digital Communications and Networks*, vol. 5, no. 1, 2018. Article (CrossRef Link)

[16] Y. Liu, J. E. Fieldsend, and G. Min, "A framework of fog computing: Architecture, challenges, and optimization," *IEEE Access*, vol. 5, pp. 25445-25454, 2017. Article (CrossRef Link)

[17] J. Grover and R. M. Garimella, "Reliable and Fault-Tolerant IoT-Edge Architecture," *2018 IEEE SENSORS*. IEEE, pp. 1-4, 2018. Article (CrossRef Link)

[18] Y. Shi, G. Ding, H. Wang, H. E. Roman, and S. Lu, "The fog computing service for healthcare," in *Proc. of 2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare*, pp. 1-5, 2015. Article (CrossRef Link)

[19] G. Manogaran, R. Varatharajan, D. Lopez, P. M. Jumar, R. Sundarasekar, and C. Thota, "A new architecture of Internet of Things and big data ecosystem for secured smart healthcare monitoring and alerting system," *Future Generation Computer Systems*, vol. 82, pp. 375-387, 2018. Article (CrossRef Link)

[20] Y. Yin, Y. Zeng, Z. Chen, and Y. Fan, "The internet of things in healthcare: An overview," *Journal of Industrial Information Integration*, vol. 1, pp. 3-13, 2016. Article (CrossRef Link)

[21] P. Verma and S. K. Sandeep, "Fog assisted-IoT enabled patient health monitoring in smart homes," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1789-1796, 2018. Article (CrossRef Link)

[22] K. Wang, Y. Shao, L. Xie, J. Wu, and S. Guo, "Adaptive and Fault-tolerant Data Processing in Healthcare IoT Based on Fog Computing," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 1, pp. 263-273, 2018. Article (CrossRef Link)

[23] T. N. Gia, A. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fault tolerant and scalable IoT-based architecture for health monitoring," in *Proc. of 2015 IEEE Sensors Applications Symposium*, pp. 1-6, 2015. Article (CrossRef Link)

[24] M. Amoon, "A framework for providing a hybrid fault tolerance in cloud computing," in *Proc. of 2015 Science and Information Conference*, pp. 844-849, 2015. Article (CrossRef Link)

[25] K. Ganga and S. Karthik, "A fault tolerent approach in scientific workflow systems based on cloud computing," in *Proc. of 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering*, pp. 387-390, 2013. Article (CrossRef Link)

[26] S. H. Gunawi, M. Hao, R. O. Suminto, A. Laksono, A. D. Satria, J. Adityatam, and K. J. Eliazar, "Why does the cloud stop computing?: Lessons from hundreds of service outages," in *Proc. of the Seventh ACM Symposium on Cloud Computing*, pp. 1-16, 2016. Article (CrossRef Link)

[27] P. Dasand and P. M. Khilar, "VFT: A virtualization and fault tolerance approach for cloud computing," in *Proc. of 2013 IEEE Conference on Information & Communication Technologies*, pp. 473-478, 2013. Article (CrossRef Link)

[28] S. Malik and F. Huet, "Adaptive fault tolerance in real time cloud computing," *2011 IEEE World Congress on Services*, pp. 280-287, 2011. Article (CrossRef Link)

[29] L. Wu, B. Liu, and W. Lin, "A dynamic data fault-tolerance mechanism for cloud storage," in *Proc. of 2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies*, pp. 95-99, 2013. Article (CrossRef Link)

[30] S. B. Baker, W. Xiang, and I. Atkinson, "Internet of Things for Smart Healthcare : Technologies, Challenges , and Opportunities," *IEEE Access*, vol. 5, pp. 26521-26544, 2017. Article (CrossRef Link)

[31] S. Han, S. Zhao, Q. Li, C. H. Ju, and W. Zhou, "PPM-HDA: Privacy-Preserving and Multifunctional Health Data Aggregation with Fault Tolerance," *IEEE Transations on Information Forensics and Security*, vol. 11, no. 9, pp. 1940-1955, 2016. Article (CrossRef Link)

[32] J. Gómez, B. Oviedo, and E. Zhuma, "Patient Monitoring System Based on Internet of Things," *Procedia Computer Science*, vol. 83, pp. 90-97, 2016. Article (CrossRef Link)

[33] G. Paré1, K. Moqadem, and C. St-Hilaire, "Clinical Effects of Home Telemonitoring in the Context of Diabetes, Asthma, Heart Failure and Hypertension: A Systematic Review," *Journal Medical Internet Research*, vol. 12, no. 2, 2010. Article (CrossRef Link)

[34] E. Perrier, Positive disruption: Healthcare, ageing and participation in the age of technology, The Mckell Institue, 2015. Article (CrossRef Link)

[35] K. J. Modi and N. Kapadia, "Securing Healthcare Information over Cloud Using Hybrid Approach," *Advances in Intellignet Systems and Computing*, vol. 714, pp. 63-74, 2019. Article (CrossRef Link)

[36] Y. Wang, M. Hu, Q. Li, X. P. Zhang, G. Zhai, and N. Yao, "Abnormal respiratory patterns classifier may contribute to large-scale screening of people infected with COVID-19 in an accurate and unobtrusive manner," *arXiv preprint arXiv*, 2020. Article (CrossRef Link)

[37] B. S. Babu, K. Srikanth, T. Ramanjaneyulu, and I. L. Narayana, "IoT for Healthcare," *International Scinece and Research*, vol. 5, no. 2, pp. 322-326, 2016. Article (CrossRef Link)

[38] R. Alexandre, O. Postolache, and P. S. Girao, "Physical Rehabilitation based on Smart Wearable and Virtual Reality Serious Game," in *Proc. of IEEE International Instrumentation and Measuremetn Technology Conference*, pp. 1-6, 2019. Article (CrossRef Link)

[39] Y. Bian, C. Yang, F. Gao, X. Sun, X. Meng, and Y. Wang, "A Physiological Evaluation Model for Flow-Experience in VR Games: Construction and Preliminary Test," in *Proc. of 2015 International Conference, Identification, Information, and Knowledge in the Internet Things*, pp. 244-249, 2016. Article (CrossRef Link)

[40] R. K. Verma, K. K. Pattanaik, S. Bharti, and D. Saxena, "In-network context inference in IoT sensory environment for efficient network resource utilization," *Journal of Network and Computer Applications*, vol. 130, pp. 89-103, 2019. Article (CrossRef Link)

[41] Z. Ahmad, A. I. Jehangiri, M. Iftikhar, A. I. Umer, and I. Afzal, "Data-oriented scheduling with dynamic-clustering fault-tolerant technique for scientific workflows in clouds," *Programming and Computer Software*, vol. 45, pp. 506-516, 2019. Article (CrossRef Link)

[42] N. Verba, K. M. Chao, J. Lewandowski, N. Shah, A. James, and F. Tian, "Modelling Industry 4.0 based Fog Computing environments for Application analysis and deployment," *Future Generation Computer Systems*, vol. 91, pp. 48-60, 2019. Article (CrossRef Link)

[43] R. Mahmud, F. L. Koch, and R. Buyya, "Cloud-fog interoperability in IoT-enabled healthcare solutions," in *Proc. of the 19th International Conference on Distributed Computing and Networking*, vol. 32, pp. 1-10, 2018. Article (CrossRef Link)

[44] R. Mahmud and R. Buyya, "Modeling and Simulation of Fog and Edge Computing Environments Using iFogSim Toolkit," *Fog Edge Computing*, 2018. Article (CrossRef Link)

Waqar Saeed is MS Scholar at Department of Information Technology, Hazara University, Mansehra, Pakistan. His research interest is Fog computing and Internet of Things.

Zulfiqar Ahmad is a lecturer in the Department of Information Technology, Hazara University, Mansehra, Pakistan. He also pursuing a PhD in Computer Science at Department of Information Technology, Hazara University, Mansehra, Pakistan. He has received his MSc (Computer Science) degree with distinction from Hazara University in 2012 and MS (CS) degree from COMSATS University, Abbottabad, Pakistan in 2016. His research interest is Fog Computing, Cloud Computing, Internet of Things, High Performance Computing, and Scientific Workflows Scheduling and Management.

Ali Imran Jehangiri is a lecturer in the Department of Information Technology, Hazara University, Mansehra, Pakistan. He graduated from Bergische University Wuppertal Germany in 2010. He received Ph.D. degree in Computer Science from the Georg-August-Univesity Goettingen, Germany in 2015. He gained industrial experience with Service Computing working as research assistant at GWDG. He is involved in research activities dealing with parallel, Grid computing, Cloud computing and Big data. He is author of several publications in international journals, and conferences.

Nader Mohamed received the Ph.D. degree in computer science from the University of Nebraska-Lincoln, Lincoln, NE, USA. He was a Faculty Member with the Stevens Institute of Technology, Hoboken, NJ, USA, and United Arab Emirates University, Al Ain, United Arab Emirates. He also has several years of industrial experience in information technology. He is currently a Professor of computer science and information systems with the California University of Pennsylvania, California, PA, USA. His current research interests include cybersecurity, middleware, Industry 4.0, cloud and fog computing, networking, healthcare systems, and cyber-physical systems.

Arif Iqbal Umar earned his MSc (Computer Science) degree from University of Peshawar Pakistan and PhD (Computer Science) degree from BeiHang University (BUAA) Beijing P.R. China. His research interests include Data Mining, Machine Learning, Information Retrieval, Digital Image Processing, Computer Networks Security and Sensor Networks. He has supervised 07 PhD candidates and 34 MS candidates. He is author of more than 70 research publications in the leading research journals and conferences. He has at his credit 27 years' experience of teaching, research, planning and academic management. He is working as Associate Professor of Computer Science in the department of Information Technology Hazara University Mansehra. He is leading the Department as Head.

Jamil Ahmad(Senior Member, IEEE) received the M.Sc. degree in information technology from the University of Warwick, U.K., the M.S. degree in computer science from the University of Peshawar, and the Ph.D. degree in artificial neural network from the Department of Electrical and Electronics Engineering, King's College London, U.K. He is currently holding the position of the Vice Chancellor at the Hazara University Khyber Pakhtunkhwa, Pakistan. He is a Fellow of the British Computer Society, a Chartered Engineer (CEng), and a member of IET and ACM. He received grants for his research and academic projects from various organizations, including the National Information and Communication Technology (ICT) Research and Development Fund, Pakistan, the HEC- British Council Linkage Program, and the U.K. Government under PMI 2 Program. He was a recipient of the Charles Wallace Pakistan Trust Visiting Fellowship, U.K., from 2012 to 2013, and the International Visiting Leadership Program (IVLP) Fellowship, from August 2007 to September 2007, USA.