Journal of King Saud University – Computer and Information Sciences 35 (2023) 544–559

HOSTED BY

Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Priority-based task scheduling and resource allocation in edge computing for health monitoring system

Zubair Sharif [a,*], Low Tang Jung [a], Muhammad Ayaz [b], Mazlaini Yahya [c], Shahneela Pitafi [d]

[a] Computer and Information Sciences Department (CISD), Universiti Teknologi PETRONAS (UTP), Seri Iskandar 32610, Malaysia
[b] Sensor Networks and Cellular Systems (SNCS) Research Center, University of Tabuk, Tabuk 71491, Saudi Arabia
[c] Head IoT Automation, Petronas, Malaysia
[d] Computer and Information Sciences Department (CISD), Universiti Teknologi PETRONAS (UTP), Seri Iskandar 32610, Malaysia

ARTICLE INFO

Article history:
Received 3 October 2022
Revised 7 December 2022
Accepted 1 January 2023
Available online 4 January 2023

Keywords:
Priority-based task scheduling
Resource allocation
Mobile edge computing
Cloud computing
Smart hospitals
Healthcare monitoring system

ABSTRACT

New and innovative wearable IoT devices for health monitoring systems (HMS) have been invented one after another. However, most of these devices are resource-constrained with restricted energy and computation power. The HMS data need to be processed via mobile edge computing (MEC) to improve the response time to fulfill the latency-sensitive and computation-intensive applications and to reduce bandwidth consumption. This paper presents an efficient task scheduling and resource allocation mechanism in MEC to meet these demands in contemplating emergency conditions under HMS. We propose a priority-based task-scheduling and resource-allocation (PTS-RA) mechanism that can assign different priorities to different tasks by considering the tasks' emergency levels computed with respect to the data aggregated from a patient's smart wearable devices. The mechanism can optimally determine whether a task should be processed locally at the hospital workstations (HW) or in the cloud. This is aimed to reduce the total task processing time and the bandwidth cost as much as possible. The proposed approach is to ensure that tasks related to the emergency are given higher priorities and to run first. After the tasks' computations, results are sent to the doctor to response promptly with quick decisions. The proposed PTS-RA was benchmarked against state-of-the-art algorithms concerning average latency, task scheduling efficiency, task execution time, network usage, CPU utilization, and energy consumption. The benchmarking results are promising as PTS-RA is capable to manage the emergency conditions and is meeting the latency-sensitive tasks' requirements with reduced bandwidth cost.

1. Introduction

A wide range of smart devices and wearable gadgets such as wireless sensors, smartphones and watches, smart vehicles, home appliances and industrial products are rapidly increasing and the market for the Internet of Things (IoTs) continues to grow (Sharif et al., 2021, Beg et al., 2022). There are now more IoT connections (connected cars, smart home devices, and connected industrial equipment) than non-IoT connections (smartphones, laptops, and computers) (Har et al., 2022). Considering the progress, it is expected that by 2025 there will be more than 75 billion IoT connections (Shome and Bera 2020, Cvitić et al., 2021, Sharif et al., 2022). The accessibility and boost of such wireless devices have led to many IoT-based applications, for example, real-time healthcare monitoring, tracking, video surveillance, autonomous vehicles etc. (Hassan et al., 2018). Among these, many IoT applications demand very short response time, and some generate a large amount of data that can overload the networks (Gaouar and Lehsaini 2021). Further, the future scenario is even more worrying as the network load continues to increase, and the processing and response time of the applications interacting with the cloud is becoming heavier. Some research studies reported that bandwidth demand almost doubles yearly (Wang et al., 2017), leading to response time issues.

The IoT devices inclined to medical applications demand more data storage and intensive data processing, hence imposing more challenges (Ayaz et al., 2019, Chellasamy et al., 2022). Serious consequences are expected to surface because these challenges are

* Corresponding author.
E-mail addresses: zubair_20000285@utp.edu.my (Z. Sharif), shahneela_22000124@utp.edu.my (S. Pitafi).
Peer review under responsibility of King Saud University.

Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2023.01.001
1319-1578/

impacting the performance of time-sensitive applications due to delays/latency in the network when offloading data to the cloud for processing (Jiang et al., 2018). Increasing demand for data storage on cloud servers for medical big data processing is becoming increasingly complex: (i) the data, when communicated, are vulnerable to security and privacy issues; (ii) the communication of the continuously collected data is not only costly but also energy hungry; (iii) operating and maintaining the sensors directly from the cloud servers are non-trial tasks. Moreover, failures to comply with sensitive timelines (latency constraints) and the challenges posed by cloud infrastructure may result in loss of life and capital investments (Yimam and Fernandez 2016, Venkatesh and Eastaff 2018). The emergence of mobile edge computing (MEC), among others, is one of the solutions to address and mitigate these challenges.

Architectural wise, MEC exists between the end devices and the cloud. In MEC the data is processed at the edge of the network instead of holding it in the cloud or on a centralized data warehouse. It provides computing resources near the IoT devices (right next to the edge devices) for local storage and preliminary data processing. Further, it assists in reducing the processing load in the IoT devices which are latency constraints and required to perform computationally intensive tasks. The purpose of shifting data computation from cloud to the edge intends to improve the response time which is highly desirable for the applications like HMS, where timely decisions are critical. Doing so helps reduce the latency in real-time data processing; moreover, it consumes less bandwidth allowing the data processing to be closer to the end devices, thus avoiding unnecessary or excessive data transmission towards the cloud (Sharif et al., 2022).

Due to these advantages, MEC has benefited medical/healthcare systems in performing heavy data processing and extracting clinical features for critical medical diagnosis. Numerous sensors have been invented for health monitoring purposes. These sensors can be configured to collect data from the human body and to perform necessary data processing at the edge node for the doctors to diagnose and make critical decisions. However, this new paradigm (MEC) is not impervious to challenges and limitations. Some challenges associated with the MEC can be found in the context of resources management (task scheduling and resource allocation), network infrastructure management, heterogeneity in data and edge devices, data management, application programming interfaces (APIs), and interoperability (Shakarami et al., 2021, Huda and Moh 2022). The finite availability of the computing resources at the edge node and the latency-constrained applications lead to a new challenge dimension. Satisfying the needs of tasks according to their latency requirements is non-trivial and requires an efficient mechanism for effective task scheduling and resource allocation (Pareek et al., 2021).

1.1. Problem statement

New and innovative wearable IoT devices for healthcare monitoring and related services have been invented one after another. However, most of these devices are resource-constrained with restricted energy and computation power. As such, the HMS data need to be processed via MEC to improve the response time to fulfill the latency-sensitive and computation-intensive applications.

As the priorities of tasks in the health monitoring system are crucial hence need careful scheduling as some jobs with the stringent latency requirements should be served earlier than the others which are latency tolerant. Such as some tasks require to be prioritized based on patient health conditions. Thus, an appropriate task scheduling algorithm is required in such situation (Jiang et al., 2020). Moreover, an efficient resource allocation mechanism is another important factor in the MEC system that must be addressed because the resource availability at the MEC server is always scarce (Medeiros et al., 2021, Sharif et al., 2021).

Based on these facts, there should be a mechanism capable of managing the emergency conditions and meeting the latency-sensitive tasks' requirements. The system can decide whether a task should be processed locally at the MEC server or remotely (at the cloud) by contemplating the task's emergency levels so that the overall task processing time and the bandwidth usage (cost) can be minimized. In addition, for the emergency or accidental conditions, lessening the response and processing time are the most significant factors so that the important/vital decision(s) can be taken timely.

However, after consulting the literature, it is found that the research on the optimal priority-based task scheduling and resource allocation for the HMS system has not been sufficiently developed. Many techniques have their own limitations and are not applicable especially where the priority-based task scheduling and resource allocation is required for the healthcare industry. Further, many of them cannot efficiently decide whether a task should be processed locally or remotely as per the task requirements. Moreover, some are consuming more network bandwidths which means higher network costs and while others are facing higher latencies.

Accepting the mentioned challenge in MEC, we have proposed a priority-based task scheduling and resource allocation (PTS-RA) system which is designed for healthcare monitoring and related services where real-time responses are of critical importance. In the proposed architecture, data generated by various sensors/devices attached to the patient are transmitted to the hospital workstations (HWs) that are deployed in the hospital. The PTS-RA system assigns different priorities to different tasks based on the data collected from the patient health status contemplating the emergency levels. The proposed system decides whether a task should be processed locally (at HW) or remotely at the centralized cloud in order to minimize the total processing/response time and bandwidth consumption. Eventually, the processed results (outcomes) are sent to the doctor for further actions. The main contributions of the proposed PTS-RA are summarized as below:

- Assigning different priorities to different tasks based on the patient health status.
- Perform the priority-based task scheduling and resource allocation according to task emergency levels and decide each task's execution location.
- To reduce the task processing time and latency.
- Conserve the network bandwidth cost.

The rest of the paper is organized as follows: Section 2 discusses some of the prominent and closely related work. Section 3 presents the proposed PTS-RA framework with all the details and necessary explanations, and Section 4 debates the simulation results and the performance evaluations to provide the authentication of the presented work. Section 5 concludes this paper along with the recommendation and possible expansion of this work.

2. Related work

With the recent success of MEC paradigm, the healthcare system is expected to be most advantageous by utilizing it along with the IoTs. To get the benefit from this opportunity, immense growth in interest has been witnessed to utilize the wireless and wearable sensors to boost the performance of the healthcare industry. Various prominent efforts tried to address different challenges, majorly including task scheduling and computation offloading (Sun and Ansari 2016), the optimization of task placement (Zhao and Liu 2018), the MEC computation resource allocation (You et al.,

2016), flying edge computing, and attaining energy efficiency etc. (Uddin et al., 2021). This section explicitly analyzes some techniques that tried to improve IoT and MEC based health facilities by resolving the issues associated with task scheduling and resource allocation.

To advance the healthcare services, the authors designed a healthcare framework in (Tuli et al., 2020) through a real-time application to monitor the patients facing heart diseases. The proposed system used MEC devices incorporating deep learning to perform different operations on the patient's data effectively. The proposed method can work in diverse MEC environment, it is customizable for different user requirements and provides better QoS. Another effort was presented by authors in (Sodhro et al., 2019), which utilizes network parameters like standard deviation, delay and jitters to accomplish their targeted objectives. The proposed architecture contained the remote access facility but at the same time it faces some delays that are considered a critical issue for the medical and healthcare applications. Further, in (Oueida et al., 2018) a healthcare-based system was offered to help the queuing systems, emergency departments, and medical resources integrating MEC with the centralized cloud. The proposed system was suitable for real-life applications and reduced patient waiting time along improving the resource utilization.

To monitor a patient suffering from chronic diseases so that the data can be gathered and processed effectively, the authors presented Fog Computing-Based IoT for Health Monitoring System (FCB-HMS) (Paul et al., 2018). The technique analyzes the security and deployment issues of the fog computing layer. Further, to identify and understand the disordered voice, a method was presented in (Sirisha and Reddy 2018). According to the proposed system, a disordered voice of a patient could be interpreted by some smart sensors. For initial processing, the collected data/information was forwarded to the edge nodes and then for further computations, this could be transmitted to the centralized cloud. After the required processing on disordered voice for fetching the information, results/outcomes were sent to the consultant for therapy or treatment. This whole system was designed to detect abnormalities to identify the tone and sensitivity in a voice. According to the outcomes, these voices were classified based on detected irregularities.

Many authors have paid attention to reduce end-to-end delays and resource utilization. Considering this fact, some works are discussed here. A method was designed in (Ren et al., 2018), where the authors tried to reduce the end-to-end latency problem by allocating the resources efficiently. They also tried to solve the issue when the computation resources are limited by utilizing them efficiently. Further, in (Nguyen et al., 2018) a similar effort was presented to perform resource allocation effectively in MEC which offered resource packages and services to enhance the utilization of MEC resources in the given budget restraints. The proposed framework satisfied fairness during the resource sharing between edge service providers and the set of users.

A novel bio-inspired hybrid algorithm (NBIHA) was developed in (Rafique et al., 2019) to manage the available resources, minimize the average response time, and optimally utilize the resources with the help of effectively task scheduling. In the proposed work, the resources were assigned and managed based on the demands of incoming requests. To enhance resources utilization, authors in (Hossain et al., 2021) proposed a framework named as Scheduling-based Dynamic Fog Computing (SDFC) which includes an extra layer between the centralized cloud and fogs which helps to choose the task execution location (either the fog node or the cloud). An IoT-based scheduling technique, Hash Polynomial Two-factor Decision Tree (HP-TDT), was presented in (Manikandan et al., 2020) to improve scheduling efficiency and decrease the response time and computational overhead by categorizing the normal and emergency conditions. By allocating the resources to the requested tasks effectively, it enhances resource efficiency and this is the target of a study presented by (Bitam et al., 2018), where the authors proposed an algorithm to allocate the set of tasks to the edge or fog nodes and the cloud data center. The proposed algorithm was helpful in reducing the task execution and processing time on the edge nodes.

To improve patient data privacy, and lessen the communication latency and data traffic, a MEC-based healthcare framework was offered in (Pace et al., 2018). Similarly, a smart in-home health monitoring system was proposed by (Verma and Sood 2018) to provide distributed storage and data mining services based on the MEC concept. Further, a cost-efficient MEC-based 5G health monitoring system was offered in (Ning et al., 2020) to reduce the system-wide cost and energy consumption of health monitoring devices. The designed technique was used to regulate the transmission rates of body sensors by bandwidth allocation to minimize the overall system cost. The channel efficacy was improved due to the use of 5G communications, but there was a trade-off between the interference encountered by channel multiplexing and the restricted computation resources of MEC servers.

A diabetic patient requires continuous monitoring and this was the target of the monitoring system provided by (Sebillo et al., 2015). The proposed method was a practical e-health solution and ideal for diabetic patients belonging to rural/countryside areas. The system intended to use various computing technologies to improve healthcare services. Moreover, patients with Parkinson's illness can also be monitored while they are at home; ultimately, it reduces the risk of falls and injuries and supports monitoring the effects of dosage alterations (Miah et al., 2017).

The techniques mentioned above have their own limitations and are not applicable, especially where priority-based task scheduling is required for the healthcare industry. Some tasks in healthcare applications are required to be prioritized based on patient health conditions. Accepting the challenge, we have proposed a task scheduling and resource allocation system for the IoT healthcare platform, considering the patient health conditions and then prioritizing the tasks accordingly.

A brief summary of state-of-the-art algorithms and schemes (discussed above) is presented in Table 1, with succinct comparisons to show the strengths and weaknesses of each approach.

3. System architecture and proposed methodology

This section presents the priority-based task scheduling and resource allocation (PTS-RA) system, which is designed for health monitoring and related services where real-time responses are demanded. The proposed system decides whether a task needs to be processed locally at any hospital workstation (HW) or remotely (at the cloud) by considering the task's emergency levels so that the overall task processing time and bandwidth cost can be minimized as much as possible. Further, minimizing the processing and response time is the most crucial aspect for emergency or accidental conditions so that the essential/vital decision(s) can be taken timely.

3.1. System architecture

Fig. 1 depicts the concept of the architecture for a patient's health monitoring system who is hospitalized for the treatment. Different connected devices and wireless or mobile sensor nodes are used for monitoring and measuring the patient's health conditions. Once the required data is measured, it will be forwarded to the HWs simultaneously. Upon receiving data at any HW, emergency levels of tasks are determined (using the proposed method),

Table 1
A summary of some major techniques and algorithms proposed for the mobile edge computing paradigm.

Article	Methodology/Algorithm	Research Domain	Achievements	Limitations
(Tuli et al., 2020)	HealthFog framework for a real-time application to monitor the patients facing heart diseases	Smart Healthcare System for Heart Diseases in integrated IoT and fog computing environments	Resource utilization using IoT devices and efficiently managing the heart patients' data for medical applications.	Data security is not considered in the HealthFog framework
(Sodhro et al., 2019)	Window-based Rate ControlAlgorithm (w-RCA)	Optimizes the medical quality of service (m-QoS) in the MEC-based healthcare	Optimizing QoS in remote healthcare applications	w-RCA shows relatively more delays
(Oueida et al., 2018)	Proposed a resource preservation net (RPN) framework	Resource utilization rate and average patient waiting time were modelled and optimized.	Resource utilization was improved for medical applications.	Data optimizations negotiate
(Paul et al., 2018)	Fog Computing-Based IoT for Health Monitoring System (FCB-HMS)	Data collection and processing in an efficient way	To monitor the patients suffering from chronic diseases	Processing speed compromised
(Sirisha and Reddy 2018)	Proposed a voice disorder detection system	Voice disorder detection system using deep learning approach	The patients can convey their voice samples captured by smart sensors for early transformations.	Higher bandwidth cost
(Ren et al., 2018)	An optimal joint communication and computation resource allocation mechanism is proposed	Communication and computation resource allocation	Efficient allocation of limited computation resources	Energy consumption is not considered
(Nguyen et al., 2018)	Price-based Resource Allocation (PBRA) for MEC	MEC resource utilization	To maximizes the edge node resource utilization in the given budget restraints	Data privacy is not considered
(Rafique et al., 2019)	Proposed NBIHA algorithm	Task scheduling and resource management	To optimize resource utilization and to reduce the average response time	Higher tasks execution time
(Hossain et al., 2021)	Scheduling-based Dynamic Fog Computing (SDFC) technique	Resource utilization	To choose the task execution place either locally or the remotely	Complexity rises when the number of user requests increases
(Manikandan et al., 2020)	Hash Polynomial Two-factor Decision Tree (HP-TDT)	Task scheduling for IoT-based devices	To improve scheduling efficiency and decrease the response time	Higher cost and power consumption
(Bitam et al., 2018)	Proposed a bio-inspired optimization approach	Task scheduling	Reduces the tasks execution and processing time at the edge nodes	A trade-off between execution time and the memory required for the tasks
(Pace et al., 2018)	Proposed architecture for human-centric applications called BodyEdge	Efficient data collection for healthcare applications	To collect and process data coming from different devices to lessen the communication latency and data traffic	Security and privacy concerns are compromised
(Verma and Sood 2018)	IoT-based remote patient health monitoring system	Patient health monitoring system	Provides services of distributed storage and data mining based on the concept of MEC	Faces reliability issues
(Ning et al., 2020)	A cost-efficient in-home health monitoring system	Bandwidth allocation to minimize the system cost	To reduce the system-wide cost and energy consumption of health monitoring devices	The system was more costly
The proposed PTS-RA (this work)	Priority-based task-scheduling and resource-allocation (PTS-RA)	Priority based Task Scheduling and Resource Allocation	To reduce the task processing time and latency. Conserve the network bandwidth cost.	

and then these are prioritized accordingly. After that, task scheduling is performed as per the task requirements, and the tasks are placed into either the HW or the cloud queue.

The HWs act as edge computing servers, and our proposed PTS-RA system is embedded/implemented on them. It means each HW has two traits: (i) containing the proposed method on them, and (ii) acting as edge computing servers. The HW process the received datasets (if it can) and then sends the processing results/outcomes to the doctors/clinicians to make further decisions. In case if the HW is unable to process the data due to insufficient resources (than the required resources) or if the task is too much time taking and is not latency sensitive then these unprocessed data sets/tasks can be sent towards the centralized cloud. If the task is processed at the cloud, then the results will be sent back to HW. At the same time, we also want to minimize the task's total processing time to achieve maximum utilization of the resources at HW, save the bandwidth cost and to attain a better performance. The patient's health monitoring devices have the capabilities of storage, computation, and connection with the internet through a wireless or wired connection. We assume that in the hospital building, many HWs are deployed for data storage and computation purposes. While all the deployed HWs are identical in terms of capabilities (i.e., computing and storage capacities), and the centralized cloud has adequate computing resources and the ability to process many tasks. Our work is easily extendable when more HWs are required to add in the running system.

3.2. Problem formulation

There is a set of tasks K generated by sensors and other connected devices and received at HW in a particular period of time. We intend to shorten the overall task processing time by ensuring the tasks with higher priority to run first. Another important consideration is the maximum use of HWs computing capacity but without overloading them. Moreover, lessen the network traffic load between HW(s) and the cloud which is caused by data transmission between them. Thus, we proposed a heuristic PTS-RA to sort out the task assignment problem.

We formulate the task assignment problem among HWs and the centralized cloud according to the task requirements and emergency levels. Each HW can serve a group of health monitoring devices and their generated data (tasks). According to some

existing research (Mao et al., 2016, Wang et al., 2017, Sun et al., 2019), the initial requirements of the task ω_i can be defined and represented as a tuple $\{\mathcal{L}^{\omega_i}, \mathcal{S}_{e_i}^{\omega_i}, D\}$, where $\mathcal{L}^{\omega_i}$ denotes the required amount of computing capacity (i.e., the total number of CPU cycles) to accomplish the task $\omega_i\omega_i$ computation at any HW, $\mathcal{S}_{e_i}^{\omega_i}$ indicates the input data size of the task ω_i which is required to store and process, and D is the maximum threshold delay constraint which each task ω_i can accept as its maximum latency tolerance. Whereas each task has a degree of emergency which is Λ so D must be satisfied for each task ω_i either for HW or cloud. For instance, if $\Lambda > D$, it means the execution of task must be processed urgently. Otherwise, the task is not too urgent or delay tolerable. The notations and their meanings for this paper are presented in Table 2. We use $\chi_{e_i}^{\omega_i}$ and $\mathrm{P}_{e_i}^{\omega_i}$ to represent the assignment of the task ω_i and the allocation of CPU capacity for the task ω_i at HW e_i, respectively. Then, the tasks assignment plan λ is expressed as a set of tasks which are mapped at various HWs as Eq. (1).

$$\begin{aligned}\lambda = [&\{e_1, (\chi_{e_1}^{\omega_1}.\mathrm{P}_{e_1}^{\omega_1}, \chi_{e_1}^{\omega_2}.\mathrm{P}_{e_1}^{\omega_2}, \chi_{e_1}^{\omega_3}.\mathrm{P}_{e_1}^{\omega_3}, \cdots, \chi_{e_1}^{\omega_n}.\mathrm{P}_{e_1}^{\omega_n})\},\\ &\{e_2, (\chi_{e_2}^{\omega_1}.\mathrm{P}_{e_2}^{\omega_1}, \chi_{e_2}^{\omega_2}.\mathrm{P}_{e_2}^{\omega_2}\chi_{e_2}^{\omega_3}.\mathrm{P}_{e_2}^{\omega_3}, \cdots, \chi_{e_2}^{\omega_n}.\mathrm{P}_{e_2}^{\omega_n})\},\\ &\{e_3, (\chi_{e_3}^{\omega_1}.\mathrm{P}_{e_3}^{\omega_1}, \chi_{e_3}^{\omega_2}.\mathrm{P}_{e_3}^{\omega_2}, \chi_{e_3}^{\omega_3}.\mathrm{P}_{e_3}^{\omega_3}, \cdots, \chi_{e_3}^{\omega_n}.\mathrm{P}_{e_3}^{\omega_n})\}, \cdots,\\ &\{e_i, (\chi_{e_i}^{\omega_1}.\mathrm{P}_{e_i}^{\omega_1}, \chi_{e_i}^{\omega_2}.\mathrm{P}_{e_i}^{\omega_2}, \chi_{e_i}^{\omega_3}.\mathrm{P}_{e_i}^{\omega_3}, \cdots, \chi_{e_i}^{\omega_n}.\mathrm{P}_{e_i}^{\omega_n})\}, \cdots] \end{aligned} \quad (1)$$

We suppose that all the data related to the task ω_i is stored at one HW e_i. Then, we find the task assignment plan λ, so that the weighted average processing time η and the traffic load Ψ need to be shortened. Thus, the entire weighted processing time η for λ calculated as Eq.(2),

$$\eta_\lambda = \sum_{\omega_k \in \left(\chi_{e_i}^{\omega_k}=0\right)} \Lambda^{\omega_k}.\frac{L^{\omega_k}}{\mathrm{P}_{e_i}^{\omega_k}} + \sum_{\omega_k \in \left(\chi_{e_i}^{\omega_k}=1\right)} \Lambda^{\omega_i}.\frac{M_{e_i}^{\omega_k}}{R_{e_i}}, \forall e_i \in \mathrm{Z}, \forall \omega_i \in \omega_k \quad (2)$$

where Λ^{ω_k} denotes the urgency levels of tasks ω_k which is $(1 \le \Lambda^{\omega_k} \le 2)$, less than or equal to 1 indicates an un-urgent task and when it exceeds from 1, it means an emergency condition has started and greater value means the most urgency level. Further $\mathcal{L}^{\omega_k}$ represents how many computing resources (i.e., the total number of CPU cycles) are required to perform the tasks ω_k computation and $\mathrm{P}_{e_i}^{\omega_k}$ is used for CPU capacity assignment/allocation on HW e_i. Moreover $\mathcal{M}_{e_i}^{\omega_k}$ is used to represent the size of the data of tasks ω_k which is required to transmit from HW e_i towards cloud and $\mathcal{R}_{e_i}$ shows the accessible or available bandwidth between HW e_i and the cloud data center. While $\chi_{e_i}^{\omega_k} = 0$ signifies tasks will be processed locally (at any HW e_i) and $\chi_{e_i}^{\omega_k} = 1$ expresses that the tasks will be allocated to the centralized cloud. When a set of tasks

Table 2
Presents the notations and their descriptions.

Notation	Description
HW e_i	A single hospital workstation
K	Set of tasks generated by sensors and other connected devices
Z	Set of all hospital workstations
$\chi_{e_i}^{\omega_i}$	Assignment of task ω_i at the HW e_i
ω_i	An arbitrary task/ size of input data needs to offload
$\mathcal{L}^{\omega_i}$	The computing capacity which is required for the task ω_i computation
$\mathrm{P}_{e_i}^{\omega_i}$	Allocation of CPU capacity to task ω_i at HW e_i
λ	Tasks assignment plan for the set of tasks
ω_k	Set of tasks receive in a certain period of time
$\mathcal{M}_{e_i}^{\omega_i}$	The size of data of task ω_i which is required to transmit from HW e_i towards cloud
ϱ	Available storage capacity of HW e_i
$\mathcal{S}_{e_i}^{\omega_i}$	The size of data of task ω_i which is needed to store at HW e_i
$\mathcal{R}_{e_i}$	The currently available bandwidth between the HW e_i and the centralized cloud
Ψ_λ	Whole network traffic load of all the tasks
η_λ	Weighted processing time
Λ^{ω_i}	Emergency level of task ω_i
hp_t	A particular health metric/parameter
ρ_e	The tasks which are put into edge queue for processing
ρ_c	The tasks which are placed into cloud queue for processing
τ	Period of time
$\tau_{\omega_k}^{tr}$	Transmission time for tasks ω_k
$\tau_{\omega_k}^{queue}$	Queuing latency for tasks ω_k
$\tau_{\omega_i}^{cmpt.HW}$	Task execution/computation time at the HW
$\tau_{\omega_i}^{cmpt.CD}$	Task execution/computation time at the cloud data center
$\tau_{\omega_k}^{p.e_i}$	Total task processing time at HW e_i
$\tau_{\omega_k}^{p.cloud}$	Total task processing time at the cloud
A_r	Set of available resources at HW e_i
$\eta_{HW_{\omega_k}}$	Perform resources allocation at HW e_i to set of tasks ω_k

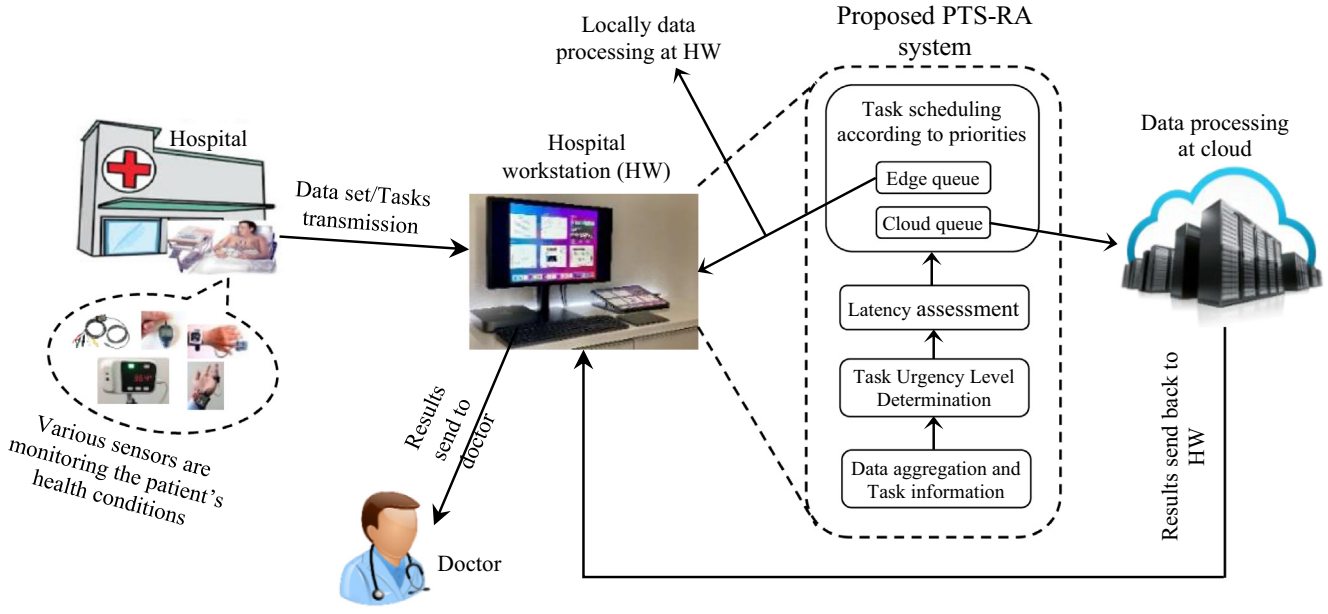

Fig. 1. The architecture of the PTS-RA scheme for the smart hospital system.

ω_k is sent from HW e_i towards the cloud data center, then the entire network traffic-load Ψ_λ for the tasks ω_k is determined by Eq. (3)

$$\Psi_\lambda = \sum_{\omega_k \in \left(\chi_{e_i}^{\omega_k}=1\right)} M_{e_i}^{\omega_k} \tag{3}$$

where $\chi_{e_i}^{\omega_k} = 1$ means the tasks are forwarded towards the cloud. As stated, we are required to lessen the entire weighted processing time and the network traffic-load. To achieve the aforementioned requirements, we formulate the task assignment problem by Eq. (4) as follows:

$$\text{Minimum of } \eta_\lambda \hat{A} \cdot \Psi_\lambda \quad \text{(to minimize both these functions)} \tag{4}$$

Constraints.

i. $\frac{s_{e_i}^{\omega_k}}{\tau} \le \varrho$ for all e_i belongs to Z and every ω_i is the element of ω_k
ii. $\frac{\mathcal{M}_{e_i}^{\omega_k}}{\tau} \le \mathcal{R}_{e_i}$ for all e_i belongs to Z and every ω_i is the element of ω_k
iii. $P_{e_i}^{\omega_k} . \chi_{e_i}^{\omega_k} \le P_{e_i}$ for all e_i belongs to Z and every ω_i is belongs to ω_k
iv. $\Lambda^{\omega_i} \in [0, 1, 2]$ for all ω_i is the element of ω_k
v. $\chi_{e_i}^{\omega_i} \in \{0,1\}$ every ω_i is the element of ω_k

In condition (i), the size of the data of tasks ω_k which is required to store at HW e_i in a unit period time which cannot exceed to the HW e_i available storage capacity ϱ, (ii) The data size $\mathcal{M}_{e_i}^{\omega_k}$ which is referred from HW e_i to the centralized cloud during a unit period time, which cannot inhabit more bandwidth than the available one. Constraint (iii) the overall computing capacity P_{e_i} (CPU utilization) of HW e_i allocated to the tasks set ω_k cannot surpass to its total/maximum usable CPU capacity. Condition (iv) signifies the task's urgency levels where $\Lambda^{\omega_i} \le 1$ means non-urgent situation and $\Lambda^{\omega_i} > 1$ means an emergency condition has started and further the greater value Λ^{ω_i} indicates its more emergency level. In the last condition (v), 0 and 1 express that the task will be allotted to any HW e_i or the centralized cloud, respectively.

3.3. Design of the PTS-RA system

Our proposed PTS-RA is a heuristic approach for solving the problem of task assignment that purposes to lessen the processing/execution time of the task and conserve bandwidth utilization while satisfying the emergency task's requirements. It determines whether a task should be assigned to a HW or the centralized cloud, and which HW will be chosen in case of local computation. Fig. 2 (flowchart) depicts how the task assignment accomplishes either on a HW or the centralized cloud.

For a new incoming task, first its urgency level is determined as explained in section *3.4*. Latency estimation or the total task processing time for both (HW and cloud) is analyzed in section *3.5*. Further in this section, the priority-based task scheduling is performed according to the tasks' emergency levels and requirements. Then the decision is made for the task computation/execution location by ensuring the shortest task processing time.

3.4. Task urgency level determination

This section explains how to assess the urgency level Λ of a task ω_i based on the collected data from the sensors and other connected devices with the patient. Whereas Λ is a parameter to describe a particular task's emergency level, which would influence the task scheduling mechanism as either a task should be executed at HW e_i or at the centralized cloud. The urgency level is identified by the sensed data collected by health monitoring devices. In our research, we considered as usual ranges of human body sensed or collected data by using the sensors of Table 3, but other than these parameters can be added according to the hospital or patient requirements.

We create the reference range for each feature on the collected data from a patient by considering the upper and lower limits of the above-mentioned parameters. We considered *T*-distribution to find (estimate) the range which is commonly used in the healthcare sector. For example, by considering the *T*-distribution for BT, r is the size of sample feature and n is a normal sample size where n is a positive number then mean $\bar{r}$ is calculated by Eq. (5)

$$\bar{r} = \frac{sumofallsamplefeatures(r_i)}{samplesize(n)} \tag{5}$$

Then the sample standard deviation (S.D) of r is determined by Eq.(6):

$$S_r = \sqrt{\frac{\sum_{i=1}^{n} (r_i - \bar{r})^2}{n-1}} \text{(S.D)} \tag{6}$$

In such a way, the upper range u_b and lower range l_b can be calculated by infimum and supremum formulas by using Eq. (7) and Eq. (8) respectively, which are written as:

$$u_b = \bar{r} - \sqrt{\frac{n+1}{n}} . s_r . t_{\alpha,n-1} \tag{7}$$

$$l_b = \bar{r} + \sqrt{\frac{n+1}{n}} . s_r . t_{\alpha,n-1} \tag{8}$$

where $t_{\alpha,n-1}$ expresses the standard coefficient for *T*-distribution for sample size n. According to the provided gender and age details of a person, the reference range of a specific health metric being supervised/monitored within l_b and u_b indicates the usual health conditions, otherwise, it will be considered an unusual or emergency condition. The recorded value of a specific health metric is represented by hp_t at a given time t. Then Λ at the provided time t can be determined by Eq.(9):

$$\Lambda = \left| \frac{(u_b - hp_t)^2 - (l_b - hp_t)^2}{(u_b - l_b)^2} \right| \tag{9}$$

When $\Lambda \le 1$ s.t$\forall$ $\omega_i \in \omega_k$ (unurgent condition).
When$\Lambda > 1$ s.t$\forall$ $\omega_i \in \omega_k$ (an emergency condition.)

Urgency level defines by the value of Λ, greater value signifies a higher degree of emergency and vice versa. More specifically, when the value of Λ is less than or equal to 1, which indicates the unurgent condition and when the value of Λ surpasses to 1 (i.e., $\Lambda > 1$), it is considered as an emergency condition has started, so the priorities will be assigned accordingly. Supposedly, in a few cases the value of Λ exceeds to 2, in order to consolidate the value when $\forall\ \Lambda > 2$, we bound the highest value by altering $\Lambda = 2$, which is the maximum urgent or emergency condition.

By assuming an example of the human BT, the reference range is l_b (lower bound) = 35.5 °C (°C) and u_b (upper bound) = 37.5 °C. If the BT (i.e., hp_t) is 36.5 °C, which means $\Lambda = 0$, representing that it is the most un-urgent condition. If the body temperature exceeds from 37.5 °C, then it means the emergency condition has started. For instance, if BT is 37.6 °C, then $\Lambda = 1.10$, which indicates an emergency condition. Another example is that if BT is 38.0 °C, then $\Lambda = 1.5$, which also signifies an urgent condition but with more emergency levels. For the most severe/serious case, if BT approaches to 39.5°Celsius, in that situation $\Lambda = 2.5$ (or some

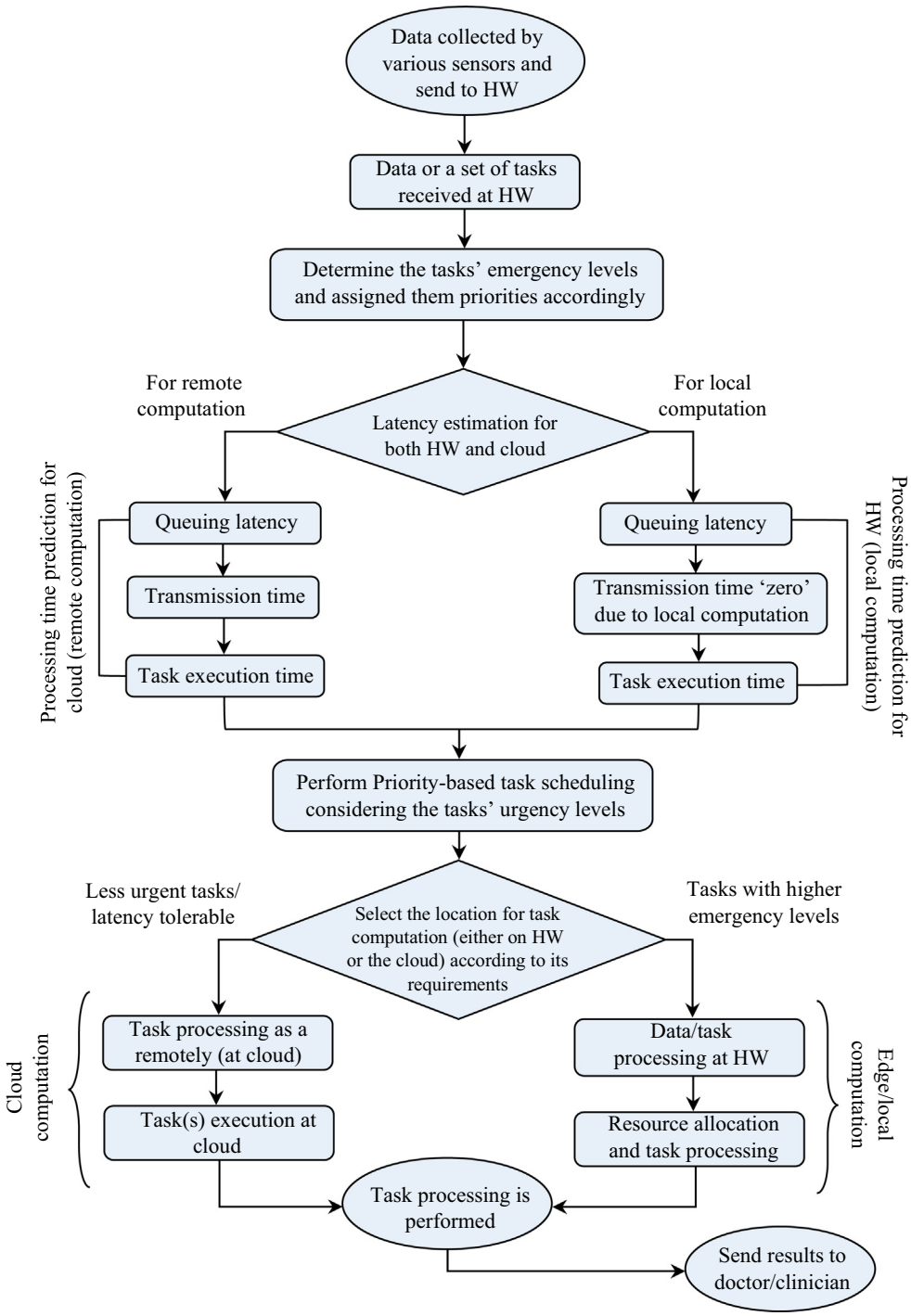

Fig. 2. Flowchart demonstrates how the task assignment is performed to a HW or the centralized cloud.

Table 3
Health monitoring features and relevant monitoring devices.

Monitored health data	Monitored from (Body sensor type)
Body temperature (BT)	Temperature sensor/ thermometer
Blood pressure (BP)	BP monitor
Blood oxygen	Pulse oximeter sensor
Blood glucose (BG)/ Blood sugar level (BGLs)	Glucose meter/glucometer
Heartbeat rate	Electrocardiogram (ECG)

hyperpyrexia cases even a human BT can touch or exceeds to 41.0, in these cases when $\Lambda > 2$ then we consolidate it to 2), which represents the highest emergency condition. In this case, all the emergency tasks are required to be executed urgently. Similarly, BP level (measured in millimeters of mercury mm Hg) can be verified by Eq. (9), we consideredl_b = 80 andu_b = 120 mm Hg, which indicates a normal health condition. For instance, when BP (i.e., hp_t) = 100 then Λ = 0.05, which shows normal condition. If the BP level decreases to 80 or exceeds to 120 mm Hg, then it will show an emergency condition. For example, if the BP is 121 or 78 mm Hg, then Λ = 1.05 and Λ = 1.1, respectively, representing

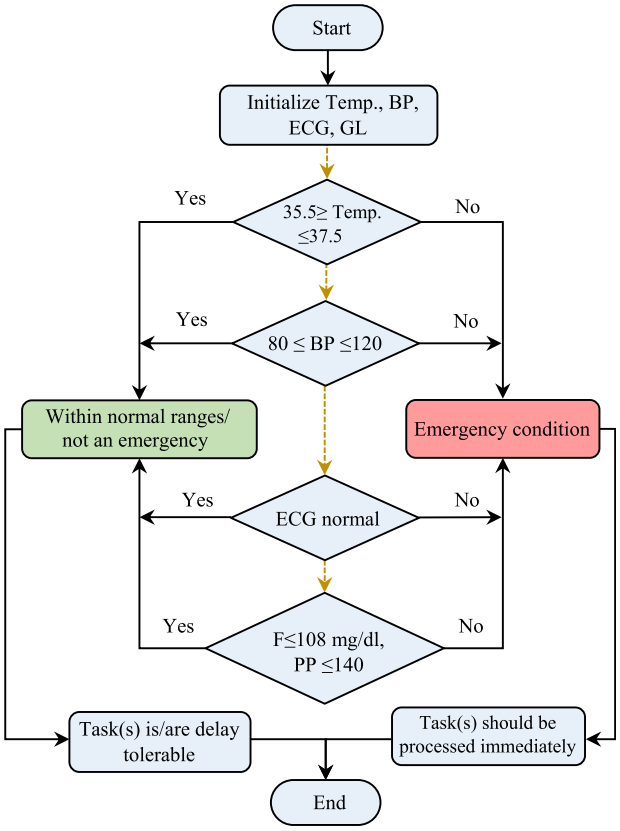

Fig. 3. Flowchart for emergency determination.

the urgent conditions. Fig. 3 (flowchart) illustrates the above-mentioned task emergency level estimation mechanism.

3.5. Latency assessment and task scheduling

In this section, the total processing time (latency estimation) for both HW and the cloud is calculated, and then the task scheduling is performed accordingly. The decision can be made whether the task should be processed at HW or cloud as per the task's requirements or with the shortest processing time.

First, the task processing time for local computation (at the HW) is considered. There is a set of tasks ω_k, which health monitoring devices transmit to the HW. To accomplish the tasks processing, there are some latencies which influence on the total processing time (represented by $\tau_{\omega_k}^{p.e_i}$) of tasks set which are: (i) time taken during the task offloading (transmission time $\tau_{\omega_k}^{tr}$ which is zero due to local computation), (ii) queuing latency $\tau_{\omega_k}^{queue}$ (iii) task execution/computation time $\tau_{\omega_i}^{cmpt.HW}$ at HW e_i, and (iv) downlink rate $dn_{\omega_k}^{delay}$ (after the task execution when the results will be downloaded), which will also be zero like data transmission time due to local computation. To calculate the above-mentioned time delays, firstly, transmission time which is expressed in Eq. (10), and then queuing latency $\tau_{\omega_k}^{queue}$ of task ω_k in the edge queue ρ_e (that tasks which will be processed at the HW e_i) is determined as in Eq. (11):

$$\tau_{\omega_k}^{tr} = \frac{\mathcal{M}_{e_i}^{\omega_k}}{\mathcal{R}_{sensor}} \tag{10}$$

s.t $\tau_{\omega_k}^{tr} \rightarrow$ neglected due to local computation

$$\tau_{\omega_k}^{queue} = \sum_{i=1}^{n} \tau_{\omega_i}^{cmpt.HW} \text{ and every } \omega_i \text{ exists in } \rho_e \tag{11}$$

where ω_i means any single task in the tasks set ω_k and n is the number of tasks proceeding in the ρ_e queue. The task computation time is inversely proportional to the processing ability of HW e_i computing server or how much computation proportion $P_{e_i}^{\omega_i}$ (i.e., CPU cycles per second) that HW e_i is allocated to the task ω_i for its execution. Then the task computation time $\tau_{\omega_i}^{cmpt.HW}$ can be calculated by Eq. (12).

$$\tau_{\omega_i}^{cmpt.HW} = \frac{P_{required}^{\omega_i}}{P_{e_i}^{\omega_i}} \tag{12}$$

This $\tau_{\omega_i}^{cmpt.HW}$ computation time is for a single task, where it can be related to the set of tasks ω_k as $\tau_{\omega_k}^{cmpt.HW}$. The sum up of all aforementioned time delays (latencies) at any HW e_i are equal to the total task processing time $\tau_{\omega_k}^{p.e_i}$ for tasks set ω_k, and it can be analyzed from Eq. (11) and Eq. (12).

$$\tau_{\omega_k}^{p.e_i} = \tau_{\omega_k}^{queue} + \tau_{\omega_k}^{cmpt.HW} \tag{13}$$

By utilizing the mentioned calculation in Eq. (13), we can compute the estimated the total processing time $\tau_{\omega_k^{'}}^{p.e_i}$ for a related task set $\omega_k^{'}$ for any other HW that does not have the information about the approximated processing time, means it will be considered as a historical log for the tasks set $\omega_k^{'}$.

Like the total task processing time $\tau_{\omega_k}^{p.e_i}$ at HW, it also exists for cloud computing which consists of a few time approaches i.e., (i) time consumed during the tasks set offloading $\tau_{\omega_k}^{tr}$ (transmission time from HW e_i to cloud data center) (ii) queuing latency when the PTS-RA will put the tasks set into ρ_c queue (the tasks which will be sent towards the cloud data center), and the cloud queuing latency $\tau_{\omega_k}^{queue}$ (when the cloud will put the tasks into the execution queue), (iii) task computation time $\tau_{\omega_k}^{cmpt.CD}$ at cloud, and (iv) downlink rate $dn_{\omega_k}^{delay}$.

The centralized cloud has much more resources than the edge node (or HW), so it has abundant computing slots for tasks computations and computing power is much greater than the HW (Kansal et al., 2022). Thus, task waiting time can be ignored within the cloud data center. When the PTS-RA puts the tasks set into ρ_c, the queuing latency will be equal to as mentioned in Eq.(14):

$$\tau_{\omega_k}^{queue} = \sum_{i=1}^{n} \tau_{\omega_i}^{tr}, \text{ and each task } \omega_i \text{ belongs to } \rho_c \tag{14}$$

In tasks set ω_k, n is the number of tasks proceeding in ρ_c. The data transmission time for the task ω_i can be computed by considering Eq. (15):

$$\tau_{\omega_i}^{tr} = \frac{\mathcal{M}_{e_i}^{\omega_i}}{\mathcal{R}_{e_i}} \tag{15}$$

where $\mathcal{R}_{e_i}$ denotes the currently available bandwidth between the HW e_i and the centralized cloud whereas $\mathcal{M}_{e_i}^{\omega_i}$ represents the amount of data required to be transmit. This transmission time is for a single task ω_i, but we can relate it to a set of tasks ω_k as $\tau_{\omega_k}^{tr}$. Further, the estimated computing time at the centralized cloud $\tau_{\omega_i}^{cmpt.CD}$ can be computed by using Eq. (16).

$$\tau_{\omega_i}^{cmpt.CD} = \frac{P_{required}^{\omega_i}}{P_{cloud}^{\omega_i}} \tag{16}$$

where $P_{required}^{\omega_i}$ is the required CPU capacity for the task ω_i and $P_{cloud}^{\omega_i}$ is the computation capacity that the cloud allocates to task ω_i during its execution. Or in other words, the computation delay at the cloud $\tau_{\omega_k}^{cmpt.CD}$ for tasks set ω_k can be expressed as $\tau_{\omega_k}^{cmpt.CD} = \frac{\tau_{\omega_k}^{cmpt.CD}}{\Xi}$, where Ξ is the computing ability difference ratio between the HW e_i and the centralized cloud. Similarly, this single task ω_i can be related to a set of tasks ω_k as $\tau_{\omega_k}^{cmpt.CD}$. According to many researchers (Chen et al., 2015, Sun et al., 2017, Yao and Ansari 2018, Zhang et al., 2018, Zou et al., 2020, Jin et al., 2021, Liao et al., 2021) we can ignore the downlink rate $dn_{\omega_k}^{delay}$ for task's results back (computation outcomes) to the HW because the data size after task computation usually is much smaller than the task before computation. Further, the download rate is much higher than the uplink rate between HWs and cloud datacenter. As there is a high-speed wired connection (called backhaul link) between them, due to very fast or high-speed wired connection, it is not influenced or becomes the cause of latency/delay for the small amount of data. Finally, we compute the estimated total processing time $\tau_{\omega_k}^{p.cloud}$ at the centralized cloud for tasks ω_k by considering Eq. (17), which is the sum of Eq. (14), Eq. (15), and Eq. (16), i.e., data transmission time, queuing time, and execution/computation time:

$$dn_{\omega_k}^{delay} neglected$$

$$\tau_{\omega_k}^{p.cloud} = \tau_{\omega_k}^{queue} + \tau_{\omega_k}^{tr} + \tau_{\omega_k}^{cmpt.CD} \quad (17)$$

As the tasks' priorities and latency estimations (for the local and remote computation) have been determined, next is to put the tasks into queues according to their requirements. So, the tasks which are in emergency situations means with higher priorities will be put into the edge queue ρ_e for the local computation at

Algorithm 1: Priority-based task-scheduling and resource-allocation (PTS-RA)

1: **Input:** Set of tasks $\omega_k = \{\omega_1, \omega_2, \omega_3, \ldots, \omega_n\}$
2: Variables Λ, K, ρ_e, ρ_c, A_r
3: **Initialization:**
4: **for** tasks ω_k received **do** // Tasks received at HW
5: **for** each ω_i **do,** where $\omega_i \in \omega_k$
6: Determine Λ from Eq. (9) // Calculate the task emergency level
7: **if** $\Lambda \leq 1$ **then**
8: unurgent condition // Within normal ranges
9: **else** $\Lambda > 1$ **then**
10: an emergency condition // hp_t is surpassed or lowered to u_b or l_b respectively
11: **end if**
12: Prioritize tasks ω_k according to the emergency levels Λ
13: **end for**
14: **for** Latency estimation and task scheduling **do** // For both task processing locations (HW and cloud)
15: **for** Calculate $\tau_{\omega_k}^{p.e_i}$ **do** // Total processing time for tasks ω_k at HW
16: $\tau_{\omega_k}^{p.e_i} = \tau_{\omega_k}^{queue} + \tau_{\omega_k}^{cmpt.HW}$ by utilizing Eq. (13)
17: **end for**
18: **for** Compute $\tau_{\omega_k}^{p.cloud}$ **do** // Total processing time for tasks ω_k at cloud
19: $\tau_{\omega_k}^{p.cloud} = \tau_{\omega_k}^{queue} + \tau_{\omega_k}^{tr} + \tau_{\omega_k}^{cmpt.CD}$ using Eq. (17)
20: **end for**
21: Perform priority-based tasks scheduling // According to tasks priorities/emergencies and latencies
22: $\rho_e \longleftarrow \sum_{i=1}^{n} \omega_i$ // Emergency tasks placed into the queue ρ_e
23: $\rho_c \longleftarrow \sum_{i=1}^{n} \omega_i$ // Less urgent tasks put into the queue ρ_c
24: **end for**
25: **for** Perform tasks processing **do** // Task execution according to the task's requirements
26: **if** ρ_e **then** /* Perform resource allocation to ω_k for their
27: $\sum_{i=1}^{n} \omega_i \left(\sum_{i=1}^{n} ar_i \ \eta_{HW_{\omega_k}} \right), ar_i \in A_r$ processing which exist in ρ_e*/
28: After the execution of task ω_i, **goto** line number 39
29: **else** ρ_c **then**
30: cloud $\longleftarrow \rho_c$ // Latency tolerable tasks are directed towards cloud
31: **if** a task ω_i is waiting in the queue ρ_c for a longer time, **then**
32: assigned a higher priority for its initiation
33: **end if**
34: **if** a task ω_i contains shorter datasets with higher queuing time, **then**
35: would have a greater priority to be run earlier
36: **end if**
37: Processing results are sent to HW
38: **end if**
39: Processing results/outcomes are sent to the doctor
40: **end for**
41: **end for**
42: **Output:** PTS-RA is performed, and the results/outcomes are sent to the doctor

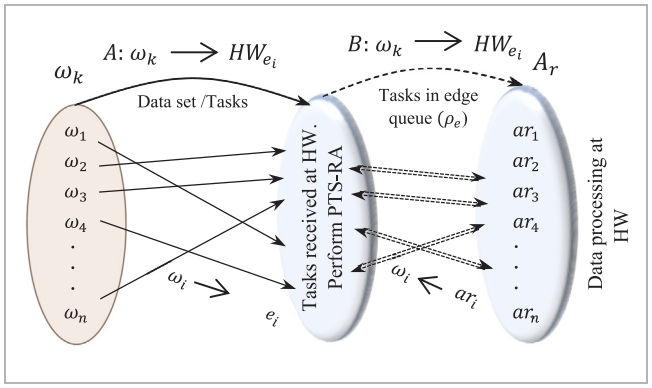

Fig. 4. Demonstrates the resource allocation for data/task processing at the HW.

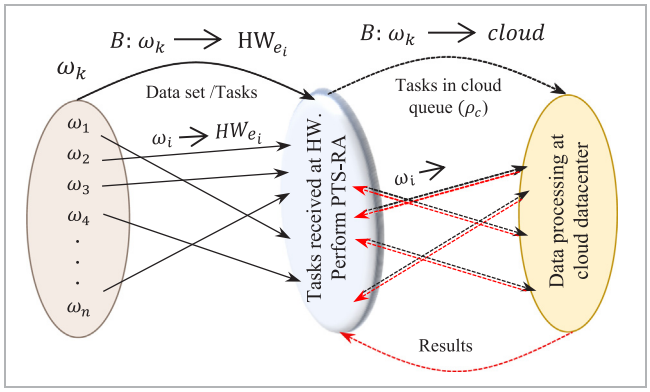

Fig. 5. Illustrates the tasks which are sent towards clouds.

HW, and the tasks which are less urgent or due to unavailability of resources at HW will be placed in the cloud queue ρ_c for remotely task processing/computation. The PTS-RA will perform it as per the task requirements or according to the emergency levels which will ensure the shortest overall task processing time and save bandwidth cost.

During task queuing, rather than by simply implementing the first-in-first-out (FIFO) method or some other mechanisms like shortest job first (SJF) scheduling, the tasks in each queue are managed according to the descending order considering their urgency levels. The newly arrived task will be placed into the queue depending on its priority value (Λ) where emergency tasks will be processed immediately as much as possible, hence lead to satisfy the emergency handling criteria of the healthcare system. Moreover, in the case of same urgency levels in local computation queue ρ_e, distinct weights can be assigned to different tasks considering various factors. This strategy will not only assist to satisfy the need for emergency handling but also helps to minimize the tasks processing time and to improve the throughput.

Algorithm 1 (*Priority-based task-scheduling and resource-allocation (PTS-RA)*).

Finally, the tasks which have been scheduled at HW_{e_i} for processing, the available resources A_r will be allocated to them for their computations by using the function in Eq. (18) and further it is expressed in Fig. 4. In this figure, left side shows the tasks ω_k are receiving at HW_{e_i} and right side of the diagram illustrates the available resources A_r (by using the proposed PTS-RA method) are responding to tasks ω_k to perform their executions. Whereas $\omega_i \leftarrow ar_i$ means an arbitrary ar_i is allocating to a random task ω_i. Further, C1 constraint signifies that the total resources allocation to the tasks set ω_k at HW_{e_i} cannot be exceeded to the maximum available resources A_r at HW_{e_i}.

$$f : A_r \rightarrow \omega_k, \text{ by fullfilling } D \tag{18}$$

$$\textit{where} \quad f(ar_i) = (\omega_i) \;, \quad \omega_i \in \omega_i \;\&\; ar_i \in A_r$$

$$\text{s.t C1} : 0 \leqslant \eta_{HW_{\omega_k}} \leqslant A_r$$

When the tasks ω_k cannot be processed locally due to insufficient or unavailability of resources at HW_{e_i} and the task is latency tolerable, then the PTS-RA system will offload the tasks ω_k to the centralized cloud as expressed in the expression or Eq. (19). Unlike the tasks in queue ρ_e, the tasks in the cloud queue ρ_c need to be transmit towards the centralized cloud for computations, which face some latencies due to wireless and wired transmissions (explained above). In this way, if a task with a bigger dataset commences transmission earlier than those with shorter datasets, the other tasks would be delayed. However, if a task has been waiting in the queue for a longer time, it will be assigned a higher priority for its initiation. Otherwise, tasks with bigger datasets or lesser emergency levels would be postponed for a longer time. Further, the task in ρ_c which contains shorter datasets with higher queuing time would have a greater priority to be run earlier. For this purpose, higher weights can be assigned to these task(s) in such situations. Moreover, to deal with these non-emergency tasks with the same urgency levels in the cloud queue ρ_c, the different weights be allocated to different tasks considering the various factors. This strategy will help to lessen the overall task processing time by improving the throughput. After putting the tasks in the queue ρ_c, finally they will be transmitted towards the cloud, which is represented in Fig. 5, where black head arrows with solid lines mean that tasks ω_k which are received at HW_{e_i}. Whereas right black arrows with dashed lines represent those tasks which are directed towards the centralized cloud, which is indicated as $B: \omega_k \rightarrow cloud$. Further, the red arrows with dashed lines indicate that the results/outcomes are sent back to HW_{e_i} from cloud after the tasks' processing.

$$B : \omega_k \rightarrow cloud \tag{19}$$

Algorithm 1 performs the PTS-RA based on tasks' emergency levels in the HMS, where the tasks' emergency levels are assessed and prioritized accordingly. The total processing time for both task execution locations (HW and cloud) is evaluated and then the PTS-RA is performed according to task urgency levels with satisfying the latency constraints tasks. For tasks execution, resources are allocated to them in case of local computation, or these are sent to the cloud for remote computation. Finally, the received processing results are sent to the doctor. The tasks from the health monitoring devices are considered as inputs and by performing the PTS-RA on them are labelled as outputs.

4. Results and discussions

Extensive simulations were performed to assess the performance of the proposed mechanism. Each HW has its own processing ability (CPU power, memory etc.). Allocation of the resources to the received tasks is done as per the task's priority (Λ urgency level), processing time, input data size, and maximum time delay D (which the task can bear). The network links description, simulation environment parameters or specifications, and the system properties for simulation settings are represented in Table 4, Table 5, and Table 6, respectively.

Table 4
The network links description during the configuration.

Source_node	Destination_node	Latency in ms
Gateway_router	Cloud_datadenter	400
HW	GW-router	300
Temperature sensor	HW	50
BP monitor	HW	45
Pulse oximeter sensor	HW	55
Glucose meter	HW	50
ECG	HW	45

Table 5
Simulation environment parameters.

Parameters	Edge node	Cloud node
CPU MIPS	[500,3000]	[2000,10000]
CPU utilization	0.5	2.0 (Cloud virtual machine)
RAM capacity	32 GB	64 GB

Table 6
System properties for simulation settings.

Properties	Specifications
Processor	Intel Core i7, 8th Gen
RAM	16 GB
OS	64-bit Windows 10 Pro
Tool	iFogSim

The iFogSim simulator (an open-source tool) was used for this purpose, through which Java as a programming language with eclipse editorial manager is followed. The graphical user interface (GUI) topology (constructed by using iFogsim) for the proposed approach is displayed in Fig. 6. Under this topology, different elements such as health monitoring sensors or connected devices, HWs, Gateway (GW) router, cloud node and connected links are utilized while characteristics of each entity were specified during the creation of topology.

The results of the proposed method were assessed with respect to the average latency, task scheduling efficiency, task execution time, network usage, CPU utilization, and energy consumption. While the proposed scheme was also compared with closely related algorithms and techniques, including the Advanced Signature-Based Encryption (ASE) algorithm (Shukla et al., 2021), NBIHA (Rafique et al., 2019), FCB-HMS (Paul et al., 2018), SDFC (Hossain et al., 2021), and HP-TDT (Manikandan et al., 2020) considering the applicability of each technique for the respective parameters.

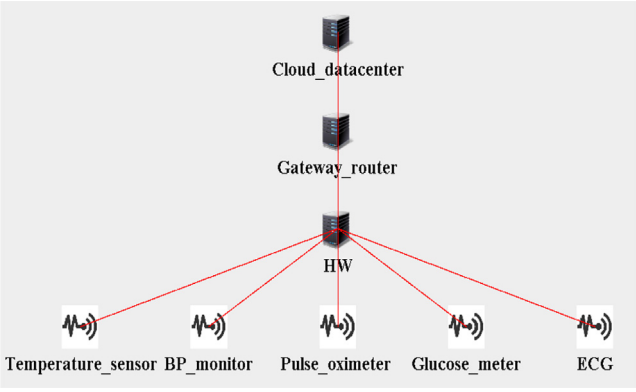

Fig. 6. GUI topology for simulation environment using ifogsim.

4.1. Average latency

The average latency comparison for HWs (by implementing the proposed approach), fog node (by using ASE technique) and the cloud is depicted in Fig. 7 when considering 0 to 200 tasks (received from different health monitoring devices and sensors). During the simulations, the maximum latency tolerance was set to

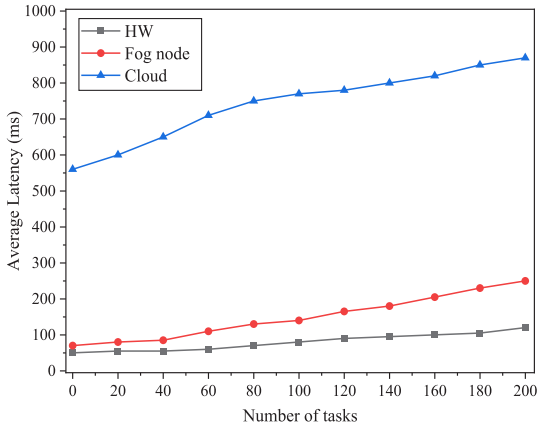

Fig.7. Average latency performance evaluation for the different number of tasks.

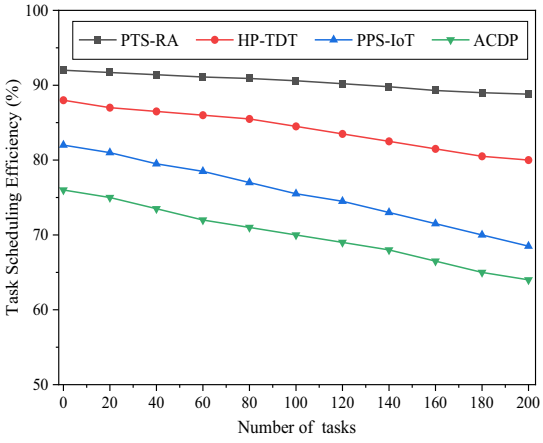

Fig. 8. Comparison of various techniques according to the task scheduling efficiency.

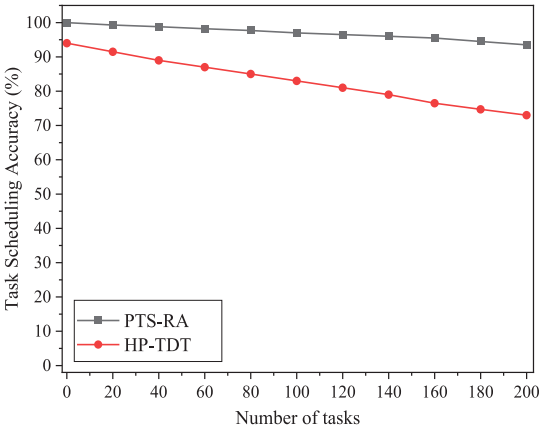

Fig. 9. Depicts the task scheduling accuracy.

150 ms and 800 ms for emergency and non-urgent tasks, respectively.

The obtained results are satisfactory at the HW according to the specified latency limits. The data processing at HWs is just like local data processing; hence, there is no need for data transmission as these are locally deployed inside the hospital. Further, the fog nodes are located a bit away from local HWs, so they provide marginally higher latency than HWs, while the cloud data centers are too far away, so they face higher latency. The simulation results reveal that the proposed scheme fulfills the tasks' maximum latency tolerance which shows its applicability, especially for emergency situations. Thus, the HW easily compete with the fog node when using the ASE scheme, and cloud in terms of maximum average latency.

4.2. Task scheduling

To assess the system's effectiveness; task scheduling efficiency, accuracy, and time were measured, and their results are presented in Fig. 8, Fig. 9, and Fig. 10 respectively. These task scheduling parameters were compared with HP-TDT, privacy-preserving smart IoT-based healthcare (PPS-IoT), and autonomic cognitive design patterns (ACDP) (Manikandan et al., 2020). During the task scheduling, the number of tasks varied from 0 to 200 and presented at the x-axis.

Firstly, the task scheduling efficiency was compared, which is demonstrated in Fig. 8. When the number of tasks increased, the task scheduling efficiency started to decrease for all methods. This is because the excessive amount of data receives on edge servers, results to overload them and leads to performance degradation. The proposed technique has an efficient mechanism for task scheduling as the tasks are scheduled and processed according to their requirements and resource availability, which helps to get better results. Similarly, task scheduling accuracy (defined as the number of tasks accurately identified and scheduled) was also evaluated and is displayed in Fig. 9, where the x-axis and y-axis show the number of tasks and the scheduling accuracy, respectively. From the given results, it can be seen that PTS-RA has more precision than HP-TDT.

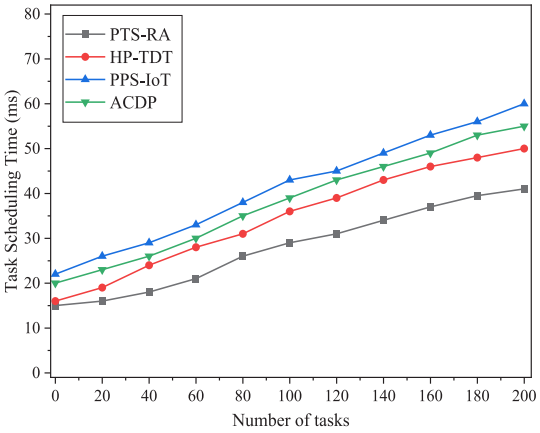

Fig.10. Illustrates the comparison with various approaches according to the task scheduling time.

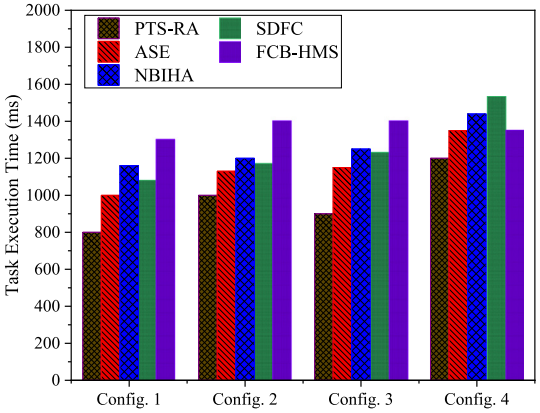

Fig. 11. Task execution time of state-of-the-art techniques considering under four various configuration settings.

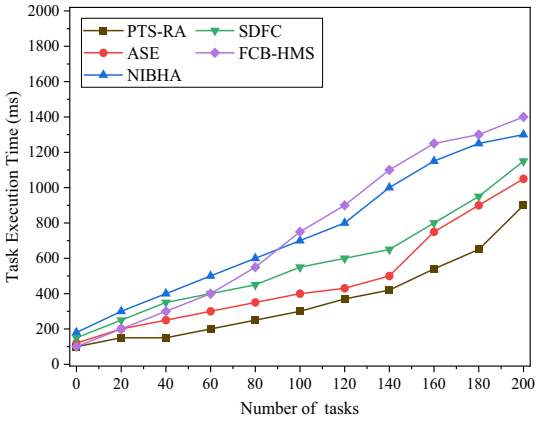

Fig. 12. An overall comparison of task execution time.

Lastly, the task scheduling time (identified as the time needed to schedule the number of tasks) of the proposed approach was tested with the baseline algorithms which is depicted in Fig. 10. Once again PTS-RA showed better performance than HP-TDT, PPS-IoT, and ACDP because it took less time for task scheduling.

Overall, the reason behind the good results is that the tasks are scheduled according to their priorities and latencies, so accordingly these are assigned at the HW or the cloud. In contrast, other techniques did not consider the task emergency levels or priorities, further these are unaware about an intelligent and efficient task and resource scheduling, so their results are compromised.

4.3. Task execution time analysis

Task execution time is another critical parameter to measure the effectiveness of any technique performing on edge computing. Considering the significance of this factor, two different evolution formations were considered to check the efficacy of the proposed method and the results are presented in Fig. 11 and Fig. 12. Initially, four distinct simulation configuration formations were considered to evaluate the task execution time of the proposed technique as shown in Fig. 11. Under each configuration, some crucial aspects like task generation time, task length, and the number of tasks (kept 100 to 200) were modified to validate the robustness of the proposed methods.

From the discovered outcomes, it can be seen that in all configuration settings, FCB-HMS is facing greater execution time in all configurations, excluding 4. Because it allocates the resources without checking their execution time, required resources, and size of input data as in many cases requests have to wait longer even if they have a petite size of data. Further, it does not deal with any mechanism for efficient resource scheduling, so it faces more execution time.

Along FCB-HMS, the NIBAH encountered with more execution time in all cases except in configuration 4, as it attempts to find

the best match of resources considering the requirements of the received tasks which ultimately surges the waiting time along with execution cost. Further, SDFC performed better than FCB-HMS and NBIHA in all configuration settings, excluding configuration 4. After that, ASE performed better than the above-discussed schemes but less than the proposed approach as in configuration 4, it faces more execution time comparatively to the FCB-HMS. Conversely, our proposed method consumed less time in all the configuration rounds than the others due to its efficient task scheduling and resource management ability which helps in reducing the processing time.

Another comparison of corresponding execution time for the number of tasks when applying the competitor techniques is represented in Fig. 12. From the simulation results, it can be seen that for all the methods, the execution time increases as the number of tasks raises. This leads to an increase in the delay, which turns to reduce the number of tasks executed/completed successfully within a specified time i.e., higher execution time means a lower number of completed tasks. In the case of FCB-HMS, at the start though it performed well even better than the SDFC and NBIHA but when the number of tasks surged then a considerable increase in its execution time is witnessed. Whereas in starting, the NBIHA has a higher execution time but after around 80 tasks it covered up its execution time and decreased it than FCB-HMS. However, ASE and SDFC have good task scheduling mechanisms, so overall they faced considerably less execution time than FCB-HMS and the NBIHA. Further, in the ASE, the resource requirement is matched from the fog node, so the fog node has sufficient processing power and less distance (than the cloud) so its execution time is shorter than SDFC.

Conversely, the execution time for the proposed scheme is significantly shorter in comparison to the mentioned techniques. As the task execution and resource allocation is performed according to the tasks' natures, either these are to be processed at HW or the cloud considering their latency constraints or emergency levels. The proposed approach assigns the latency-sensitive tasks to the HWs and computing-intensive tasks with lower emergency levels to the cloud data center. It tries to utilize the maximum computing capacity of HW by saving the network cost. Such load balancing by efficient task scheduling reduces the computational and queuing time which ultimately can attain better performance on the task processing time. Another key aspect that should be mentioned here, except the proposed approach, all other techniques entertain the tasks with the same priorities during the processing. This means the emergency tasks (with higher Λ values) may not be processed earlier than the less critical ones, which leads to higher delays for the latency constraints tasks. Whereas PTS-RA processes the tasks according to their emergency levels, the tasks with higher priorities are processed earlier than those with lower priorities. Therefore, the tasks can meet their maximum latency restraints. Thus, the proposed scheme consistently outperforms and achieves the shortest task processing time among the different methods.

4.4. Network usage

In order to regulate the maximum network usage in maintaining the cost, an efficient task scheduling is required which not only maintains a balanced cost but also avoids unnecessary network traffic (which becomes the cause of network congestion) on a communication channel. Since every task has its fixed data size for transmission, so it is easy to say that the higher the network utilization, the greater the cost. The network utilization (during the data transmission or the different number of tasks) evaluation for all methods is demonstrated in Fig. 13. Analyzing the network usage for HW, the bandwidth utilization is the lowest even almost negligible among all the paradigms due to no need of data transmission to the cloud data center. As it processes all the data/tasks locally, which saves the total bandwidth cost. Although HW attains the lowest network usage, but the computing capacity of HW is much lower than the centralized cloud so, it cannot complete all the tasks on time, especially in the case of larger data set sizes.

While in the case of fog nodes (using the ASE technique), network utilization increased with the increment in number of tasks and reached around to 300 megabytes (MBs) because of data transmission to a specific location. However, this network usage is still significantly lower than the cloud since the distributed deployment of fog nodes is over a region, so it achieved a better efficacy and performance than the cloud. However, it still cannot complete all the tasks on time especially for those emergency tasks. Whereas the network utilization for the cloud is too high even more than double to the fog node due to its too much distance from the end devices, thus the data needs to travel from source to destination for the long distances over the multi-hops via wide area network (WAN).

As a result, it causes in excessive growth of network cost due to data travelling towards the cloud and network congestion due to extreme data transmission, which sometimes prevents timely data transmission. Further, it takes a long data transmission time to transmit tasks or data packets for large sizes, resulting in high network consumption cost. In addition, some tasks are required a massive amount of data for transmission which surges the network load enormously, many times all the tasks cannot be finished on time which leads to performance degradation. In short as the number of tasks for computation offloading is increased, the network utilization is also boosted, so the network cost is highest for this paradigm.

In the case of proposed PTS-RA scheme, it allocates the tasks either to HW or the cloud based on their emergency levels, the data size, and the available bandwidth between HW and the cloud. When dealing with it, it trade-offs between network usage and the tasks compilation time; therefore it can escape the problems of too much cost or network congestion. Based on this fact, the proposed method used a particular amount of bandwidth according to the task's nature and network cost constraints to avoid network congestion which enables timely task completion. Moreover, to lessen the network load and processing time, the tasks with higher emergency levels or less amount of data for computation are assigned to the HWs. The tasks that are required larger processing resources with lower urgency levels tend to be allocated to the private cloud. Based on the proposed mechanism, it assigns some tasks to the private cloud, that's why its network cost is higher than HW. Due to that the proposed method achieves a balanced

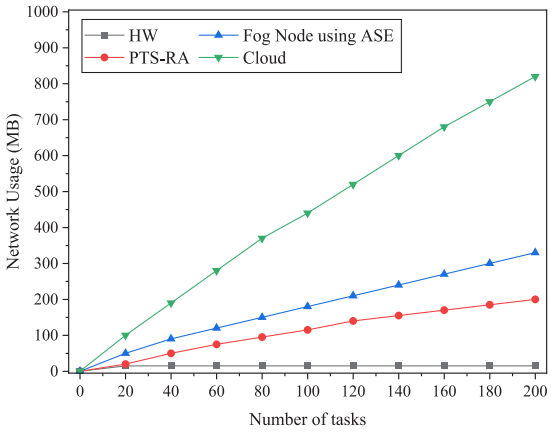

Fig. 13. Comparison of network usage across various techniques.

load on HWs and the network usage which attains a lower network cost compared with fog node while using the ASE and the cloud.

4.5. CPU utilization

CPU utilization percentage comparison for PTS-RA, ASE, and SDFC algorithms is presented in Fig. 14. In the case of SDFC and ASE, they tried to process maximum tasks locally without considering the task length, the available computing capacity, and the required computing resources. Hence, the CPU utilization is too high (almost 100 %), due to that, it cannot accommodate more tasks, which cannot guarantee the performance of the tasks' processing due to overloading.

Whereas the proposed method assigns the tasks according to their nature, e.g., urgency level, resource requirements and expected processing time at the HW and the cloud. Further, the tasks that need to be processed less amount of data i.e., require a lower number of computing resources and their priorities are also high, tend to be assigned to HW.

If the task is not latency-sensitive and has too much data for its processing, in that case the proposed system can assign the task(s) to the cloud data center which helps to reduce unnecessary task processing and helpful in achieving the optimized CPU utilization. At the same time, when the proposed algorithm assigns tasks, it considers the available computing capacity at each HW, the tasks latency estimation, the priority of task, and the task length, which means it can avoid overloading the HWs. It significantly improves the performance of the computing nodes and the execution of tasks. Therefore, the acquired results revealed a better CPU utilization performance and without overloading the HWs.

4.6. Energy consumption

Fig. 15 depicts the energy consumption comparison for the different techniques considering the various number of tasks. It can be observed that the energy consumption increases with the increment in number of tasks from 0 to 200. It is evident that as the number of tasks increases, the system requires more time for their executions which leads to higher energy consumption. Further, the excessive switching of the processor during the tasks' processing due to a large amount of data also becomes the cause of inappropriate energy consumption.

By analyzing the figure, a decision can be made about the energy consumption pattern like FCB-HMS > HBIHA > SDFC > ASE > PTS-RA e.g., the hierarchy about which these techniques consumed the energy. It is apparent that the proposed approach consumed relatively lower energy than all other methods. This is because of its efficient task scheduling and resource management ability. It executes the urgent tasks locally and sends the less emergent tasks with the more extensive data set to the cloud, which makes it energy efficient. Further, it also arranges the tasks according to their priorities so that later on it helps to prevent an unnecessary switching in finding the shortest job first (SJF) or first come first serve (FCFS) task/jobs. Thus, not blindly allocates resources without realizing the tasks demands and their natures. On the contrary, the benchmarking schemes don't have such precise mechanisms hence they face excessive energy consumption.

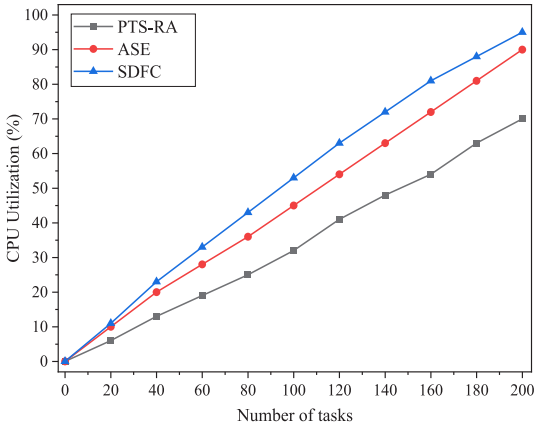

Fig. 14. CPU utilization for the different number of tasks.

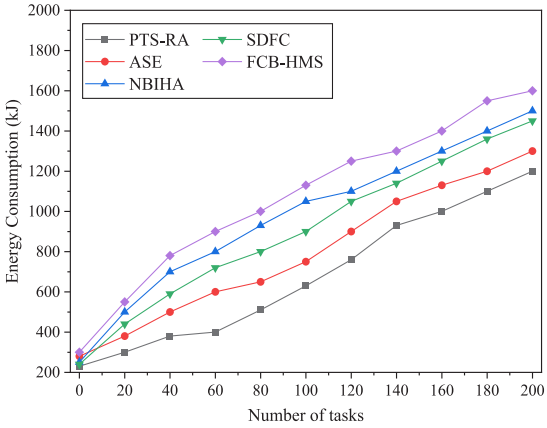

Fig. 15. Energy consumption concerning the various number of tasks.

4.7. Brief discussion

The simulation results are promising against the benchmarking algorithms and techniques as the PTS-RA can manage the emergency conditions and meet the latency-sensitive tasks' requirements with reduced bandwidth cost. The reasons behind the encouraging results are as it assigns different priorities to different tasks by considering their emergency levels. So, the tasks with higher priorities run first which satisfies the latency-sensitive tasks' requirements. Further, the proposed mechanism optimally determines whether a task should be processed locally at the hospital HW or in the cloud. For this purpose, it estimates the queuing, transmission, and computing latencies for both HW and the cloud to predict the total processing time of a task. In addition, it maintains two queues: the local tasks execution queue and the cloud tasks execution queue, while the tasks in the respective queue will be handled accordingly. This aims to reduce the total task processing time and the bandwidth cost as much as possible. By doing so, it avoids overloading any HW and reduces the network cost.

On the other hand, in most cases, the effectiveness of each technique or algorithm begins to suffer as the number of tasks increases in a comparison. The rationale for this occurrence is that system performance degrades, and response times lengthen when more tasks compete for the same computing resources. However, the proposed PTS-RA can effectively deal with such situations, leading to the overall improved performance under all circumstances.

5. Conclusion and future work

The smart healthcare technology is constantly helping to develop better services for the healthcare sector and making hospitals smarter. In this paper, we designed a priority-based task scheduling and resource allocation (PTS-RA) mechanism through the edge computing paradigm. The proposed system is intended to monitor the patient's health conditions and helps to make

timely decisions. Based on the patient health surveillance/ monitoring data, the proposed system assigns different priorities to different tasks according to their necessity condition that enables emergency tasks to be processed earlier. At the same time, it avoids to overload any hospital workstation (HW) and reduces the network cost. Further, it estimates the queuing, transmission, and computing latencies for both HW and the cloud to predict the total processing time of a task. PTS-RA maintains two queues: the local tasks execution queue and the cloud tasks execution queue, while the tasks in the respective queue will be handled accordingly. Lastly, it decides whether a task should be processed locally at the HW or the cloud according to its requirements or with the shortest processing time. After processing the tasks, the results are sent to the doctor for further action considering the patient condition. According to the simulation results, the performance of the proposed approach is better when compared with the other techniques. In future, we plan to develop a mechanism that shares the tasks among the HWs with the help of a software-defined network or by using some other mechanisms.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was conducted in Universiti Teknologi PETRONAS (UTP) under the YUTP grant scheme with the reference code of #RG2022-0754, the cost center of 015LC0-413, and the project title of "Edge Computing Oriented IoT Operation and IoT Resources Optimization".

References

Ayaz, M., Ammad-Uddin, M., Sharif, Z., et al., 2019. Internet-of-Things (IoT)-based smart agriculture: Toward making the fields talk. IEEE Access. 7, 129551–129583.

Beg, S., Handa, M., Shukla, R., et al., 2022. Wearable smart devices in cancer diagnosis and remote clinical trial monitoring: transforming the healthcare applications. Drug Discovery Today.

Bitam, S., Zeadally, S., Mellouk, A., 2018. Fog computing job scheduling optimization based on bees swarm. Enterprise Information Systems. 12 (4), 373–397.

Chellasamy, A., A. Nagarathinam and IoT, 2022. An Overview of Augmenting AI Application in Healthcare. Computer Networks, Big Data. 397-407.

Chen, X., Jiao, L., Li, W., et al., 2015. Efficient multi-user computation offloading for mobile-edge cloud computing. IEEE/ACM Transactions on Networking. 24 (5), 2795–2808.

Cvitić, I., Peraković, D., Periša, M., et al., 2021. Ensemble machine learning approach for classification of IoT devices in smart home. International Journal of Machine Learning and Cybernetics., 1–24

Gaouar, N., Lehsaini, M., 2021. Toward vehicular cloud/fog communication: A survey on data dissemination in vehicular ad hoc networks using vehicular cloud/fog computing. International Journal of Communication Systems. 34 (13), e4906.

Har, L.L., Rashid, U.K., Te Chuan, L., et al., 2022. Revolution of Retail Industry: From Perspective of Retail 1.0 to 4.0. Procedia Computer Science. 200, 1615–1625.

Hassan, N., Gillani, S., Ahmed, E., et al., 2018. The role of edge computing in internet of things. IEEE communications magazine. 56 (11), 110–115.

Hossain, M.R., Whaiduzzaman, M., Barros, A., et al., 2021. A scheduling-based dynamic fog computing framework for augmenting resource utilization. Simulation Modelling Practice and Theory. 111, 102336.

Huda, S.A., Moh, S., 2022. Survey on computation offloading in UAV-Enabled mobile edge computing. Journal of Network Computer Applications. 103341.

Jiang, Y.-L., Chen, Y.-S., Yang, S.-W., et al., 2018. Energy-efficient task offloading for time-sensitive applications in fog computing. IEEE Systems Journal. 13 (3), 2930–2941.

Jiang, C., Fan, T., Gao, H., et al., 2020. Energy aware edge computing: A survey. Computer Communications. 151, 556–580.

Jin, Z., Zhang, C., Jin, Y., et al., 2021. A resource allocation scheme for joint optimizing energy-consumption and delay in collaborative edge computing-based industrial iot. IEEE Transactions on Industrial Informatics.

Kansal, P., Kumar, M., Verma, O.P., 2022. Classification of resource management approaches in fog/edge paradigm and future research prospects: a systematic review. The Journal of Supercomputing., 1–60

Liao, Z., Peng, J., Xiong, B., et al., 2021. Adaptive offloading in mobile-edge computing for ultra-dense cellular networks based on genetic algorithm. Journal of Cloud Computing. 10 (1), 1–16.

Manikandan, R., Patan, R., Gandomi, A.H., et al., 2020. Hash polynomial two factor decision tree using IoT for smart health care scheduling. Expert Systems with Applications. 141, 112924.

Mao, Y., Zhang, J., Letaief, K.B., 2016. Dynamic computation offloading for mobile-edge computing with energy harvesting devices. IEEE Journal on Selected Areas in Communications. 34 (12), 3590–3605.

Medeiros, A., Braun, T., Di Maio, A., et al., 2021. REACT: A solidarity-based elastic service resource reallocation strategy for Multi-access Edge Computing. Physical Communication. 47, 101380.

Miah, S.J., Hasan, J., Gammack, J.G., 2017. On-cloud healthcare clinic: an e-health consultancy approach for remote communities in a developing country. Telematics and Informatics. 34 (1), 311–322.

Nguyen, D.T., Le, L.B., Bhargava, V., 2018. Price-based Resource Allocation for Edge Computing: A Market Equilibrium Approach. IEEE Transactions on Cloud Computing. 1–1. https://doi.org/10.1109/tcc.2018.2844379.

Ning, Z., Dong, P., Wang, X., et al., 2020. Mobile edge computing enabled 5G health monitoring for Internet of medical things: A decentralized game theoretic approach. IEEE Journal on Selected Areas in Communications. 39 (2), 463–478.

Oueida, S., Kotb, Y., Aloqaily, M., et al., 2018. An Edge Computing Based Smart Healthcare Framework for Resource Management. Sensors (Basel) 18 (12). https://doi.org/10.3390/s18124307.

Pace, P., Aloi, G., Gravina, R., et al., 2018. An edge-based architecture to support efficient applications for healthcare industry 4.0. IEEE Transactions on Industrial Informatics. 15 (1), 481–489.

Pareek, K., Tiwari, P.K., Bhatnagar, V., 2021. Fog Computing in Healthcare: A Review. IOP Conference Series: Materials Science and Engineering. IOP Publishing.

Paul, A., Pinjari, H., Hong, W.-H. et al., 2018. Fog computing-based IoT for health monitoring system. Journal of Sensors. 2018.

Rafique, H., Shah, M.A., Islam, S.U., et al., 2019. A novel bio-inspired hybrid algorithm (NBIHA) for efficient resource management in fog computing. IEEE Access. 7, 115760–115773.

Ren, J., Yu, G., Cai, Y., et al., 2018. Latency Optimization for Resource Allocation in Mobile-Edge Computation Offloading. IEEE Transactions on Wireless Communications. 17 (8), 5506–5519. https://doi.org/10.1109/twc.2018.2845360.

Sebillo, M., Tortora, G., Tucci, M., et al., 2015. Combining personal diaries with territorial intelligence to empower diabetic patients. Journal of Visual Languages & Computing. 29, 1–14.

Shakarami, A., Shahidinejad, A., Ghobaei-Arani, M., 2021. An autonomous computation offloading strategy in Mobile Edge Computing: A deep learning-based hybrid approach. Journal of Network Computer Applications. 178, 102974.

Sharif, Z., Jung, L. T. and Ayaz, M. 2022. Priority-based Resource Allocation Scheme for Mobile Edge Computing. 2022 2nd International Conference on Computing and Information Technology (ICCIT), IEEE.

Sharif, Z., Jung, L.T., Razzak, I., et al., 2021. Adaptive and Priority-based Resource Allocation for Efficient Resources Utilization in Mobile Edge Computing. IEEE Internet of Things Journal.

Sharif, Z., Jung, L.T., Ayaz, M., et al., 2022b. Smart Home Automation by Internet-of-Things Edge Computing Platform. International Journal of Advanced Computer Science Applications. 13 (4).

Shome, S., Bera, R., 2020. Narrowband-IoT base station development for green communication. Advances in greener energy technologies. Springer:, pp. 475–487.

Shukla, S., Thakur, S., Hussain, S., et al., 2021. Identification and Authentication in Healthcare Internet-of-Things Using Integrated Fog Computing Based Blockchain Model. Internet of Things. 15, 100422.

Sirisha, G. and Reddy, A. M. 2018. Smart Healthcare Analysis and Therapy for Voice Disorder using Cloud and Edge Computing. 2018 4th International Conference on Applied and Theoretical Computing and Communication Technology (iCATccT), IEEE.

Sodhro, A.H., Luo, Z., Sangaiah, A.K., et al., 2019. Mobile edge computing based QoS optimization in medical healthcare applications. International Journal of Information Management. 45, 308–318.

Sun, X., Ansari, N., 2016. EdgeIoT: Mobile edge computing for the Internet of Things. IEEE Communications Magazine. 54 (12), 22–29.

Sun, J., Gu, Q., Zheng, T., et al., 2019. Joint communication and computing resource allocation in vehicular edge computing. International Journal of Distributed Sensor Networks. 15 (3), 1550147719837859.

Sun, Y., Zhou, S., Xu, J., 2017. EMM: Energy-aware mobility management for mobile edge computing in ultra dense networks. IEEE Journal on Selected Areas in Communications. 35 (11), 2637–2646.

Tuli, S., Basumatary, N., Gill, S.S., et al., 2020. Healthfog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated iot and fog computing environments. Future Generation Computer Systems. 104, 187–200.

Uddin, M.A., Ayaz, M., Mansour, A., et al., 2021. Cloud-connected flying edge computing for smart agriculture. Peer-to-Peer Networking and Applications., 1–11

Venkatesh, A., Eastaff, M.S., 2018. A study of data storage security issues in cloud computing. International Journal of Scientific Research in Computer Science, Engineering and Information Technology. 3 (1), 1741–1745.

Verma, P., Sood, S.K., 2018. Fog assisted-IoT enabled patient health monitoring in smart homes. IEEE Internet of Things Journal. 5 (3), 1789–1796.
Wang, H., J. Gong, Y. Zhuang, et al., 2017. Healthedge: Task scheduling for edge computing with health emergency and human behavior consideration in smart homes. 2017 IEEE International Conference on Big Data (Big Data), IEEE.
Wang, C., Yu, F.R., Liang, C., et al., 2017a. Joint computation offloading and interference management in wireless cellular networks with mobile edge computing. IEEE Transactions on Vehicular Technology. 66 (8), 7432–7445.
Yao, J., Ansari, N., 2018. QoS-aware fog resource provisioning and mobile device power control in IoT networks. IEEE Transactions on Network Service Management. 16 (1), 167–175.
Yimam, D., Fernandez, E.B., 2016. A survey of compliance issues in cloud computing. Journal of Internet Services and Applications. 7 (1), 5.
You, C., Huang, K., Chae, H., et al., 2016. Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Transactions on Wireless Communications. 16 (3), 1397–1411.
Zhang, J., Xia, W., Yan, F., et al., 2018. Joint computation offloading and resource allocation optimization in heterogeneous networks with mobile edge computing. IEEE Access. 6, 19324–19337.
Zhao, L., Liu, J., 2018. Optimal placement of virtual machines for supporting multiple applications in mobile edge networks. IEEE Transactions on Vehicular Technology. 67 (7), 6533–6545.
Zou, J., Hao, T., Yu, C., et al., 2020. A3C-DO: A regional resource scheduling framework based on deep reinforcement learning in edge scenario. IEEE Transactions on Computers. 70 (2), 228–239.