Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 19(1): 513–536.
DOI: 10.3934/mbe.2022024
Received: 22 October 2021
Accepted: 12 November 2021
Published: 17 November 2021

Research article

Secure and failure hybrid delay enabled a lightweight RPC and SHDS schemes in Industry 4.0 aware IIoHT enabled fog computing

Mazhar Ali Dootio[1,*], Abdullah Lakhan [1,4], Ali Hassan Sodhro[2,3], Tor Morten Groenli[4], Narmeen Zakaria Bawany[5] and Samrat Kumar[6]

[1] Research Lab of AI and Information Security,Benazir Bhutto Shaheed University Lyari, Karachi, Sindh Pakistan

[2] Department of Computer Science, Kristianstad University, SE-291 88 Kristianstad, Sweden

[3] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518000, China

[4] Kristiania University College, Department of Technology, Mobile Technology Lab, OSLO, Norway

[5] Department of Computer Science and Software Engineering, Jinnah University for Women, Pakistan

[6] Charles Sturt University, Australia

* **Correspondence:** Email: Abdullahrazalakhan@gmail.com.

Abstract: These days, the Industrial Internet of Healthcare Things (IIT) enabled applications have been growing progressively in practice. These applications are ubiquitous and run onto the different computing nodes for healthcare goals. The applications have these tasks such as online healthcare monitoring, live heartbeat streaming, and blood pressure monitoring and need a lot of resources for execution. In IIoHT, remote procedure call (RPC) mechanism-based applications have been widely designed with the network and computational delay constraints to run healthcare applications. However, there are many requirements of IIoHT applications such as security, network and computation, and failure efficient RPC with optimizing the quality of services of applications. In this study, the work devised the lightweight RPC mechanism for IIoHT applications and considered the hybrid constraints in the system. The study suggests the secure hybrid delay scheme (SHDS), which schedules all healthcare workloads under their deadlines. For the scheduling problem, the study formulated this problem based on linear integer programming, where all constraints are integer, as shown in the mathematical model. Simulation results show that the proposed SHDS scheme and lightweight RPC outperformed the hybrid for IIoHT applications and minimized 50% delays compared to existing RPC and their schemes.

Keywords: IIoHT; scheduling; hybrid delay; jobs; healthcare; RPC; fault-tolerant; security

1. Introduction

These days, industry 4.0 is the new trend of the digital healthcare sector, where many traditional manual healthcare systems have been converting into digital healthcare systems [1]. For instance, many patients directly receive the services of doctors without any physical visit to the hospital system [2]. Industry 4.0 offers cyber-physical transformation of construction to promote networked manufacturing, and the digital convergence of industry, enterprises, and other processes such as physical hospitals, have been converted into digital hospitals [3,4]. The digital healthcare applications in industry 4. 0 consist of healthcare sensors, edge computing data centers, and wireless network technologies. The industrial internet of healthcare things (IIoHT) supports these applications via different services [5]. The essential infrastructure of Industry 4.0 is based on distributed computing principles, with many nodes geographically distributed and connected in networks [6, 7]. In distributed computing, all the IIoHT applications have been assumed as thin clients where their workloads are offloaded to the thick heavyweight cloud servers for execution [8,9]. However, due to the offloading of data to edge computing, there is a significant risk of data security and delay in the IIoHT network. Generally, the data offload in plaintext and face malware attacks or anomaly attacks on the data by different sources. Therefore, data security for IIoHT applications in cloud computing poses a significant challenge in the network [10]. Generally, IIoHT applications are delay-sensitive and consist of sensitive tasks such as high heartbeat emergency, high blood-pressure strain, and many [11]. Therefore, delay enabled task scheduling problem with many types of delays (e.g., computational delay, security delay, and network delay) has gained a lot of attention for each IIoHT application [12].

Many studies investigated delay optimal scheduling problems for IIoHT applications in the edge computing network [1, 7, 12–16]. Computational delay, network delay, and failure aware delay are some of the constraints that have been applied to optimal delay issues. Fine-grained, coarse-grained, and workflow workloads are all evaluated, and they are divided into heterogeneous and homogeneous edge nodes. The considered workloads are fine-grained, coarse-grained, and workflows and edge them into heterogeneous and homogeneous edge nodes. These applications designed the runtime for IIoHT based on the remote procedure call (RPC) mechanism. The RPC offers different socket classes to allow thin-client (mobile devices and bio-sensors) to connect to the edge nodes to execute applications. In RPC, the Javascript Object Notation (JSON) is a protocol that is implemented to offload data from mobile devices to the edge node for the computation. Many encryption and decryption based on RSA, MD5, SHA-256, and homoeomorphic techniques introduced in RPC for IIoHT applications. Furthermore, based on checkpointing and primary backup techniques, the failure-aware approaches were introduced in RPC for IIoHT applications in edge computing [17–20]. However, these studies only focused on computational delay and network delay constraints and widely ignored the security delay and failure delay of IIoHT applications in edge computing.

The hybrid delay scheduling problem for IIoHT applications at heterogeneous edge nodes is formulated in this work. Different restrictions, such as computing delay, network delay, security delay, and fault-tolerant delay, make up the hybrid delay. The goal is to reduce the total number of hybrid delays for all IIoHT applications that have deadlines. The study develops the unique RPC system and its schemes, as well as contributes to state-of-the-art research. (I) The paper presents a realistic simulation runtime RPC environment for IIoHT applications based on socket programming. Many Arudino bio-sensors are connected to mobile devices in the system. Their workload is offloaded to the JSON protocol, which edge computing uses to process them within their deadlines. (II) The study devises the secure hybrid delay scheme (SHDS) to optimize each delay, which schedules all healthcare workloads within their deadlines. To deal with the challenge, the researchers devised a mathematical model based on linear integer programming, in which all constraints are integer. The research proposes a four-phase architecture for securing fault-tolerant delay workload assignment. (III) The research creates a mathematical model for the topic at hand and simulates IIoHT applications using the system's constraints.

Summary, the study formulates the hybrid delay enabled scheduling problem for IIoHT applications in heterogeneous edge computing. The considered problem is closely related to existing studies [1, 7, 11, 15, 19, 20] where these studies considered the network delay and computational delay for IIoHT in heterogeneous edge nodes. Based on computational delay and network delay, these studies designed the RPC based on socket programming and JSON protocols. However, our work is different from existing studies in which we consider more delay constraints such as computational delay, network delay, security delay, and fault-tolerant delay in heterogeneous edge computing. This study designed the lightweight and hybrid delay enabled RPC system and its schemes for IIoHT applications and simulated with the mathematical model in the considered problem.

The manuscript has the subsequent sections. Section 2 investigated the prevailing coarse-grained and fine-grained workload execution within the distributed network. Section 3 shows the problem-solution based on the proposed architecture and Section 4 shows algorithm implementation based on heuristics steps. After the algorithm framework devised. Section 5 determined the performances of the proposed schemes with the considered problem. The Section 6 summaries the effort of the proposed work with the achievement of the results.

2. IIoHT RPC related work

In this part, the study discusses the efforts of existing studies for delay-optimal task scheduling for IIoHT in edge computing. The studies [1–4] investigated computational enabled offloading based on RPC mechanism inhomogeneous edge nodes for coarse-grained IIoHT applications. The goal was to minimize computational delay for all applications. The workload is coarse-grained in which virtual machine-based RPC runtime executes them with minimum delay. However, these studies only focused on the computational delay for IIoHT applications. The studies [5–10] investigated hybrid constraints such as network delay and computational delay for IIoHT applications in homogeneous edge nodes. The objective is to minimize offloading network delay and assignment computational delay of all applications. The applications are bio-sensors that enable fine-grained and offload the proximity edge node for computation. Whereas network delay and the computational delay were the constraints during the minimization of the objective function in the study. The studies [11–15] suggested hybrid delay-optimal workload assignment in heterogeneous edge cloud. Where proximity cloudlets are placed near user applications and remote cloud-deployed away from the user applications. The resource constraint and computational delay were assumed to be the constraint of applications and minimized the overall delay. However, the wait delay, network delay, and failure delay were ignored in the study. These studies [16–20] formulated the network delay and resource delay in the edge computing for IIoHT applications. The goal was to minimize resource and network offloading delays of applications in the distributed heterogeneous edge nodes in the network. These studies implemented the coarse-grained healthcare workload of applications and full offloading schemed designed in the RPC system.

These studies [21–27] formulated the hybrid delays (resource renting delay and computational delay) and suggested scheduling schemes in heterogenous mobile edge cloud paradigms for fine-grained healthcare applications. The goal is to minimize the renting and computational delays of resources of healthcare applications. The dynamic task offloading, and scheduling schemes were suggested to deal with mobility and intermittent changes in resources for the applications. The studies [28–32] suggested the microservices-based lightweight computational delay enabled resources for the IIoHT applications. These studies devised the lightweight microservices-based RPC to run the IIoHT applications with minimum boot-up delay and execution delay in edge computing. These studies [33–37] suggested container enabled RPC to minimize services delay, boot up delay and computational delay of IIoHT applications in edge computing. The goal was to minimize the hybrid delay of applications in distributed edge nodes in the network.

To the best of our knowledge, the hybrid delay with failure, security, network, and computational delays for IIoHT in distributed heterogeneous edge nodes has not been studied yet. The existing RPC considered the boot-up delay, network delay, service delay, and network delay of IIoHT applications. However, failure, network, secure and computational delay of IIoHT applications are still to be improved further in this study.

3. Problem description

The study devises the hybrid delays optimal RPC enabled edge computing for IIoHT system that consists of client and management components as shown in Figure 1. The IIoHT client component comprises biomedical sensors, network status, servers status, and results interfaces. Each biomedical sensor generates data for the fine-grained healthcare tasks, and the workload offloader engine decides when to offload them to IIoHT management for execution. The workload offloader engine is a method that is located at think client and makes the offload decision and displays the results in the mobile interfaces. How do tasks works with the edge nodes in the system? The proposed approach has been designed based on RPC runtime where stub classes of client tasks communicate with edge node skeleton classes for the execution. The biomedical tasks are represented by $t = 1, \dots N$. IIoHT management component consists of computing infrastructure where RPC skeleton runtime is implemented to run the tasks stub client tasks. IIoHT management handles all requested generated from the client-side and process them according to their attributes. IIoHT management has two layers such as SHDS runtime layer and the SHDS framework layer. The SHDS runtime is an infrastructure where IIoHT clients communicate with edge nodes for applications execution based on the RPC mechanism. The runtime denotes fog-1 and fog-2. These two edge computing nodes are used for processing client socket to the server socket. The client socket is mobile devices, and the server socket is edge nodes designed based on the RPC mechanism and communicate via JSON protocol to process data between them. Each edge node is represented by $k = 1, \dots, K$, and has different speeds and resources for executing tasks $t = 1 \in N$. In order to satisfy the requirements of all tasks on

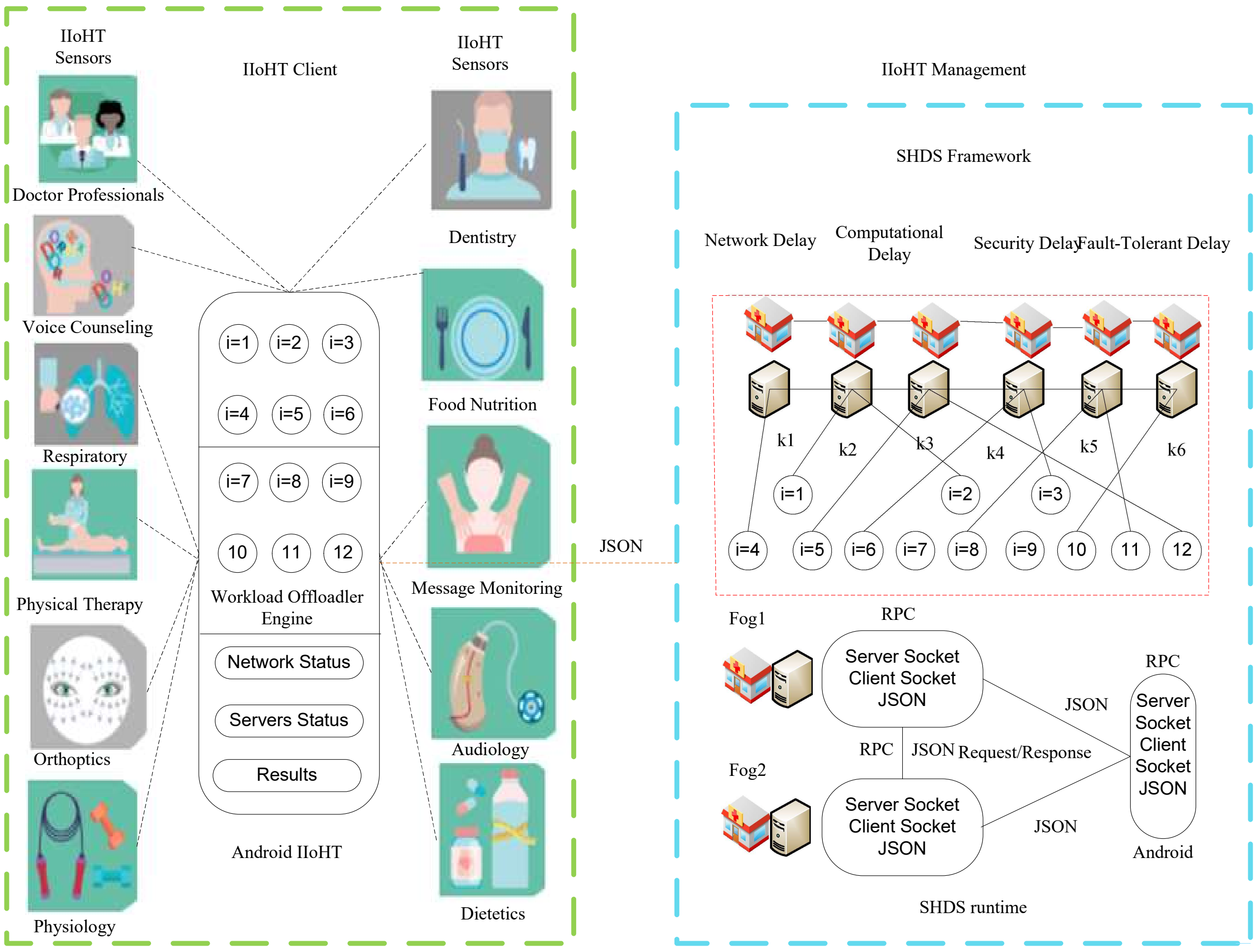


Figure 1. Hybrid delay optimal bio-sensors connected RPC enabled edge computing IIoHT system.

heterogeneous edge nodes, the study devises the secure hybrid delay scheduling (SHDS) framework, which consists of network delay, security delay, computational delay, and network delay heuristics. These fours heuristics optimize the four constraints of IIoHT tasks in the heterogeneous nodes in the system.

3.1. Problem formulation

The consider architecture as shown in Figure 1 consists of N number of biomedical fine-grained tasks, e.g., $N\{t = 1, \dots, N\}$. Each task has deadline d_t and workload w_t. The edge computing consists of homogeneous K node, e.g., $K = \{k = 1, \dots, K\}$. Each edge node k has limited space, e.g., ϵ_k. All nodes have homogeneous computing speed ζ_k in distributed computing. The study considers the base stations $\{b = 1, \dots B\}$, which are communication channel for application to offload their tasks to the edge computing.

Table 1. Mathematical notation.

Notation	Description
N	It represents the total number of tasks
t	Particular task of N
w_t	Workload of task t
d_t	Deadline of task t
K	Number of homogeneous edge nodes
k	The node k of K
ϵ_k	The resources of node k
B	Number of base stations
b	Particular base station of B
b_C	The channel capacity
B_{bw}	The bandwidth of the channel in hertz
b_S	Signal power over the bandwidth
N_s	Interference over the bandwidth
$\frac{S}{N_s}$	The signal-to-noise ratio

3.2. Computational delay

The study exploits binary assignment variables for workload assignment on edge computing. For instance $x_{t,k} = \{0, 1\}$. If the $x_{t,k} = 1$, the workload is assigned to the computing node k; otherwise, it becomes 0. The study calculates the computational delay in the following way.

$$Com = \sum_{t=1}^{N} \sum_{k=1}^{K} \frac{w_t}{\zeta_k} \times x_{t,k}. \tag{3.1}$$

Equation (3.1) shows the computational delay of all tasks on all computing nodes.

3.3. Network delay

The study considers the network delay between IoV applications and edge computing. The study exploits binary assignment variable for workload offload via available base stations. For instance $y_{t,b} = \{0, 1\}$. If the $x_{t,b} = 1$, the workload is offloaded to available base station b, otherwise it becomes 0. The study determines the computational delay in the following way.

$$Net = \sum_{t=1}^{N} \sum_{b=1}^{B} \frac{w_t}{b_w} \times y_{t,b}. \tag{3.2}$$

Equation (3.2) shows the network delay of all tasks offload to base-stations. Whereas, b_w is the bandwidth of wireless channel in the edge computing network.

3.4. Security delay

The study implements the RSA asymmetric scheme based on 256 bits to perform encryption and decryption in edge computing. The study determines the security delay of all tasks in the following way.

$$Sec = \sum_{t=1}^{N}\sum_{k=1}^{K} enc(t,k,pk) + dec(t,k,pv)(\times x_{t,k}. \tag{3.3}$$

Equation (3.3) shows the security delay of all tasks on all computing nodes. Whereas, *pk* and *pv* variables are 256 bits primary key and private key are keys in asymmetric RSA scheme which is implemented in our model.

3.5. Fault-tolerant delay

Each task has binary status such as $z_{t,k,s} = \{0, 1\}$. The notation $z_{t,k,s} = 1$ shows the task finished successfully in the system, otherwise failure task will gain zero status. Furthermore, the fault-tolerant delay of tasks delimits in the following way.

$$FT = \sum_{t=1}^{N}\sum_{k=1}^{K} \times z_{t,k,s} = 0\neg 1. \tag{3.4}$$

Equation (3.4) determines the fault-tolerant delay of tasks in the system.

The objective function of the study is to minimize the delay of all tasks and determine in the following way.

$$Z = Com + Net + Sec + FT. \tag{3.5}$$

The workload assignment problem mathematically based on linear integer programming is formulated in the following way.

$$\min Z, \quad \forall t = 1, \ldots N. \tag{3.6}$$

Equation (3.6) represents the objective function which minimize the total delays of all tasks in edge computing environment based on linear integer programming.

$$\sum_{t=1}^{N}\sum_{k=1}^{K} \frac{w_t}{\zeta_k} \times x_{t,k} = 1 \leq \epsilon_k. \tag{3.7}$$

Equation (3.7) determines that all tasks must be executed under the available nodes resource and not exceed their limits.

$$Z \leq d_t, \quad \forall t = 1, \ldots, N. \tag{3.8}$$

Equation (3.8) determines that, all tasks must be executed under the their deadlines.

$$\sum_{t=1}^{N} x_{t,k} = 1, \quad \forall k = 1, \ldots K. \tag{3.9}$$

Equation (3.9) determines that, each task is assigned to one computing node at a time.

$$\sum_{k=1}^{K} x_{t,k} = 1, \quad \forall t = 1, \ldots 1. \tag{3.10}$$

Equation (3.10) determines that each node execute one task at a time.

4. Proposed algorithm SHDS framework

The study implies the secure hybrid delay scheme (SHDS), which schedules all healthcare workloads below their deadlines. To determine the optimal assignment problem, the study formulated this problem based on linear integer programming, where all constraints are integer, as shown in the mathematical model. In the proposed work, the study makes the optimal assignment with fault-tolerant and security features, and the study devises a secure fault-tolerant delay workload assignment algorithm framework that consists of four different phases. The phase-1 of Algorithm 1 acknowledges the offloading tasks of IIoHT applications to edge computing for computation, where each base station channel has sufficient bandwidth for the offloading. Suppose there is no bandwidth available in the wireless channel of bandwidth or weak signal with high inference. In that case, the IoV application can not offload tasks to the edge node for the computation. After the successful offloading to the edge node, phase-2 of Algorithm 1 performs encryption and decryption on fully homomorphic encryption based on RSA asymmetric technique with 256 bits key length such as primary key and private key in any computing node. The main goal is to encrypt and decrypt data to the only node, and others can execute the workload on encrypted form and send it back to the original computing node. At the same time, phase-3 of Algorithm 1 searches the optimal edge node for the computation on the offloaded workload under their deadlines. In phase-4 of Algorithm 1 monitors the running tasks is they failed during execution, then Algorithm 1 will call all phases and search another node until and unless the workload has a deadline. If the fault-tolerant is timeless than the remaining deadline of the task, Algorithm 1 will successfully recover the failure task by using the primary backup strategy of fault-tolerant in edge computing nodes. In this way, the overall delays can handle in the optimal in the proposed Algorithm 1.

4.1. Security delay

After the successful offloading phase, edge computing takes the input of tasks data, e.g., t_w. The input of the IoV application will be encrypted first at the initial edge computing node by applying fully homomorphic encryption based RSA 256-bits length. The security is asymmetric where the primary key is exploited for the encryption, and the private key is exploited for the decryption at the same edge computing node. The study devises the fully homomorphic encryption enabled technique in phase as defined in Algorithm 2. The study generating the public key length and private key length keys based on RSA key generator class [32]. The list[t,k] is the encrypted list of all tasks based on Algorithm 2. The security rules are designed based on the homomorphic scheme, which is more time-efficient and has more minor security delays than existing all traditional security schemes for IIoHT tasks. The conventional security schemes are MD5, SHA-256, CRC-32, and others, and dual processes such as encrypt and decrypt each data on every node and cant apply computation on the ciphertext. This is a time-consuming process as well as consumes much more resources of nodes. Figure 2 shows the mechanism of encryption and decryption in the system. The study introduces lightweight encryption and decryption-enabled cryptography asymmetric approach. All tasks encrypt and decrypt on at any node and storge their encrypted data on the system storage. In our process, initially, the system will select the node for encryption and decryption with public and private keys to avoid security delay in the system. To ensure security and delay, the study encrypts the data only once on one node, and another node can apply computation on encrypted data instead of plaintext. After the execution, all results can

Algorithm 1: SHDS Framework

```
Input  : $B, N, K$
begin
    Level-1;
    foreach (b=1 to B) do
        if ($y_{t,b}$ == 1) then
            Calculate the Network Delay Based on Eq (3.2);
            Level-2;
            Encrypted and Decrypted all tasks based on Eq (3.3);
            Function ($enc = RSA.Enc(t1) \leftarrow k1, pk$);
            Function ($dec = RSA.Dec(enc) \leftarrow k1, pv$);
        else
            Wait for communication channel if $y_{b,t} = 0$;
    Level-3 foreach (k=1 to k& t=1 to N) do
        if ($x_{t,k}$ == 1) then
            Calculate the computation Delay Based on Eq (3.1);
            if $Com.t1 \leq d_t$ & $w_t \leq \epsilon_k$ then
                Search the optimal node $k^*$ for assignment;
                $Z^* \leftarrow t \leftarrow k$;
        else
            Wait for resources to be free of nodes if $x_{k,t} = 0$;
    Level-4;
    if ($y_{t,k,s}$ == 0) then
        Calculate the fault-tolerant based on Eq (3.4);
        if ($x_{t1,k1} w_t \leq k_2 \& Comp.t1.k1 \neg comp.t1.k2 \leq d_t$) then
            Move $x_{t1,k1,s} = 0$ to $x_{t1,k2,s} = 1$;
        else
            Recover all failures under their deadlines;
            $Z^* \leftarrow Z$;
    End Inner Loop;
End Main;
```

be decrypted by a particular node in which tasks first encrypted their data during processing.

4.2. Network delay

In the first phase, the study calculates the network delay before offloading workload to edge computing. Generally, network delay is a complex task due to many factors such as available channel capacity of base stations b_C, bandwidth B_{bw}, signal strength S, inference noise N between IoV device

Algorithm 2: Phase-2 Security Delay Scheme

```
Input  : $(c^e = (t1, t2, \ldots tN)\ mod\ n)$;
Output: $(m = c^d\ mod\ n)$ ;
1  begin
2      $p \leftarrow$ large integer;
3      $q \leftarrow$ large integer;
4      $p \neq q$;
5      $n \leftarrow p \times q$ ;
6      $\phi(n) = (p-1) \times (q-1)$;
7      Choose an integer $\epsilon$ when $1 < \epsilon\phi(n)$;
8      foreach $(t = 1, \ldots, N\&\ k = 1, \ldots .K)$ do
9          $list[t,k] \leftarrow enc(t,k,pk \times \phi)(\times x_{t,k})\frac{t_w}{\zeta}$;
10         Offload list to available nodes;
11         $dec(t,k,pv)(\times x_{t,k})$;
12     Finish all encryption;
13     foreach (Result[t,k]) do
14         $dec(t,k,pv)(\times x_{t,k}) \leftarrow Result[t,k]$;
15     Display Results;
16 End loop;
```

and edge computing. We determine the channel capacity of base stations in the following way.

$$b_C = b_{bw} log_2(1 + \frac{S}{N}) \times Net. \tag{4.1}$$

Equation (4.1) determines the capacity of all base stations in edge computing network.

$$Net = \sum_{t=1}^{N} \frac{W_t}{b_{bw}} \times List[t,k]. \tag{4.2}$$

Equation (4.2) determined the network delay for each task during offloading the data to the servers for execution. Whereas, due to results, still downloading data incurs some delay as determined in Eq (4.2). The threshold and condition matched in the following way.

$$Threshold_b \leq b_C. \tag{4.3}$$

Equation (4.3) determines the threshold quality of service of offloading in the network. The offloader engine in devices offload data and download results with the minimum amount of delay whenever data required for the processing in the system.

4.3. Computational delay and priority

Algorithm 3 is a scheduler based on linear integer programming-based scheduling greedy where all tasks have priority and deadline constraints. The priority and deadline are assigned to each task

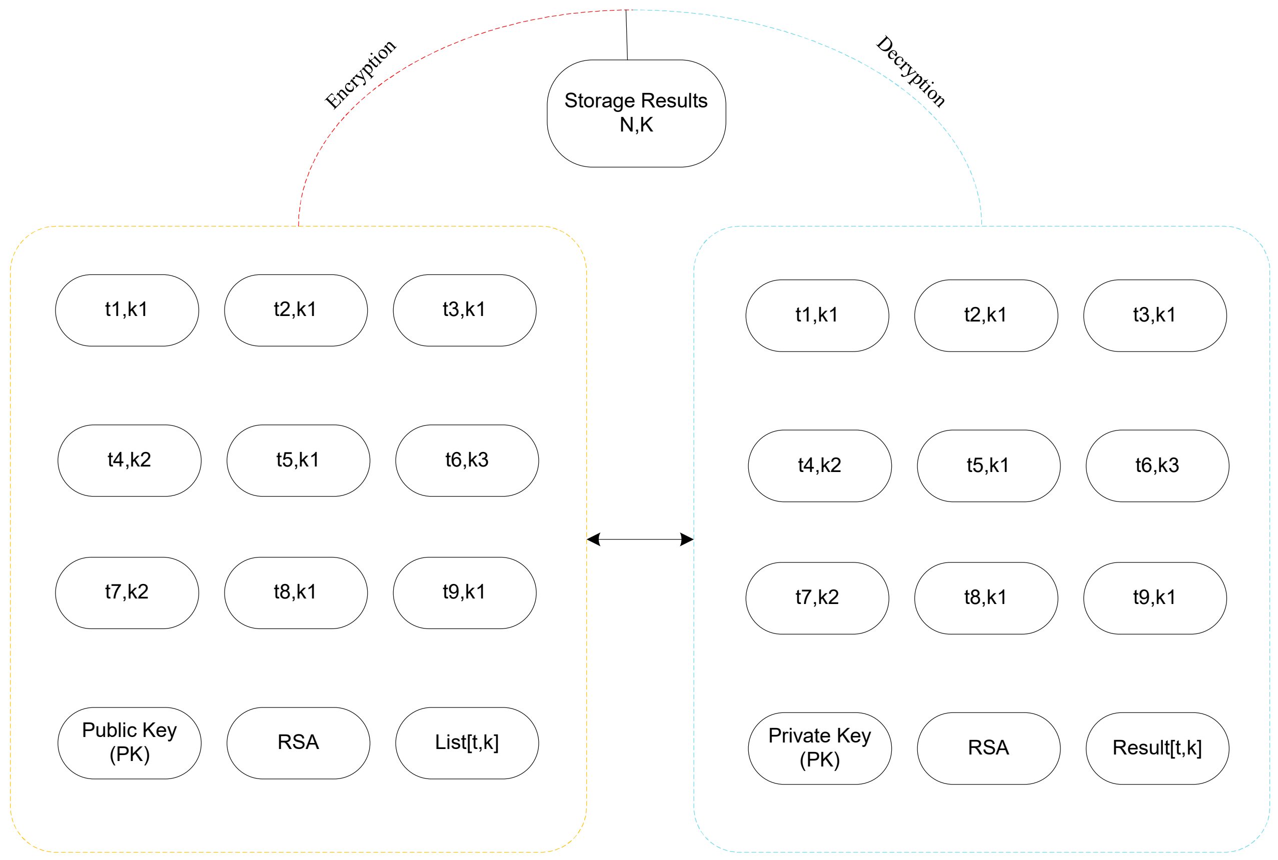


Figure 2. Encryption and decryption in system.

during design time by users. Therefore, the scheduler does only workload assignments of tasks to the available computing nodes with their resources. Algorithm 3 sorts all tasks by priority, such as high priority tasks must schedule first, then low priority tasks. However still, high-priority tasks have critical deadlines too. Therefore, Algorithm 3 sort all tasks by their deadlines based on the earliest deadline first (EDF) [26]. To ensure scheduling, Algorithm 3 sort all edge computing nodes by their resource capacity. Algorithm 3 takes encrypted tasks as an input and performs their computation based on their deadlines.

The output of Algorithm 3 shown in Table 2. All tasks are scheduled according to their deadlines without missing any deadline as shown in Table 2.

4.4. Fault-tolerant delay

The fault-tolerant scheme in the paper works like the process helper in which failure of tasks can reschedule from the point of failure as shown in Algorithm 4. Many existing fault-aware frameworks include checkpointing, primary backup files technique, migration, and masking techniques. The fundamental goal of these techniques is to offer a robust and seamless environment to run applications without degradation their performances. In this work, the study modified the checkpointing and primary backup techniques [38, 39] to handle the failure delay in the system. The primary goal is to minimize failure delay of tasks whenever they have failed during scheduling in the system. Figure 3

Algorithm 3: Computational delay scheme

```
   Input  : $(list[t,k] = enc(t,k,pk \times \phi)$);
1  begin
2      foreach $(t, k \leftarrow list[t,k])$ do
3          sort all tasks by their priorities Priority Scheme;
4          sort all tasks by their deadlines based EDF;
5          sort all edge computing nodes by their storage;
6          Apply scheduling on ciphertext;
7          if $Com \leftarrow t1 \leftarrow k1 \leq d_t \& \ t_w \leq \epsilon_k$ then
8              Apply scheduling on ciphertext;
9              $enc(t,k,pk \times \phi) \times x_{t,k}$
10         Repeat Scheduling all tasks are assigned to nodes;
11         if $(z_{t,k,s} = 0)$ then
12             Call Fault-Tolerant Method;
13             $Results[t,k] \leftarrow List - f[t,k,s] \leftarrow z_{t,k,s} = 0 \neg 1$;
14             until status convert from 0 to 1;
15 End loop;
```

Table 2. Computational delay and task priority.

N	Priority	K	Com(ms)	$d_t(ms)$	$\epsilon_k(GB)$
t_1	high	k1	100	110	10GB
t_2	high	k2	400	490	15GB
t_3	low	k3	500	1100	20GB
t_4	low	k4	1000	1200	30GB
t_5	high	k1	1500	2000	7GB
t_6	high	k3	50	60	15GB
t_7	high	k2	30	60	8GB
t_8	high	k4	25	50	5GB
t_9	high	k1	88	110	6GB

shows the scheduling process of tasks onto the heterogeneous fog nodes under the deadline and assignment constraints. The blue node shows that a task t is assigned to the fog node k for the execution. All the tasks are assigned based on the deadline of tasks onto the available nodes. The purple color shows that the assigned tasks on nodes met their deadlines during scheduling in the system. If the two tasks have arrived at the same node, then according to their deadline, they are scheduled on to the node; if a task is already executing on the node, then the next task will wait until the completion of the first task execution. For instance, $t3, k4$ is already running as shown in the green-covered area, and then task $t6, k4$ will wait until finish time $t6$ during scheduling. If the task failed as shown in Figure 3 with yellow node, and its status changed from status 1 to 0. The scheduler will call Algorithm 4 to recover tasks with the checkpointing technique on the primary backup scheme in the network. For instance, a task $t2$ failed on node $k2$, but it has a deadline. It recovers

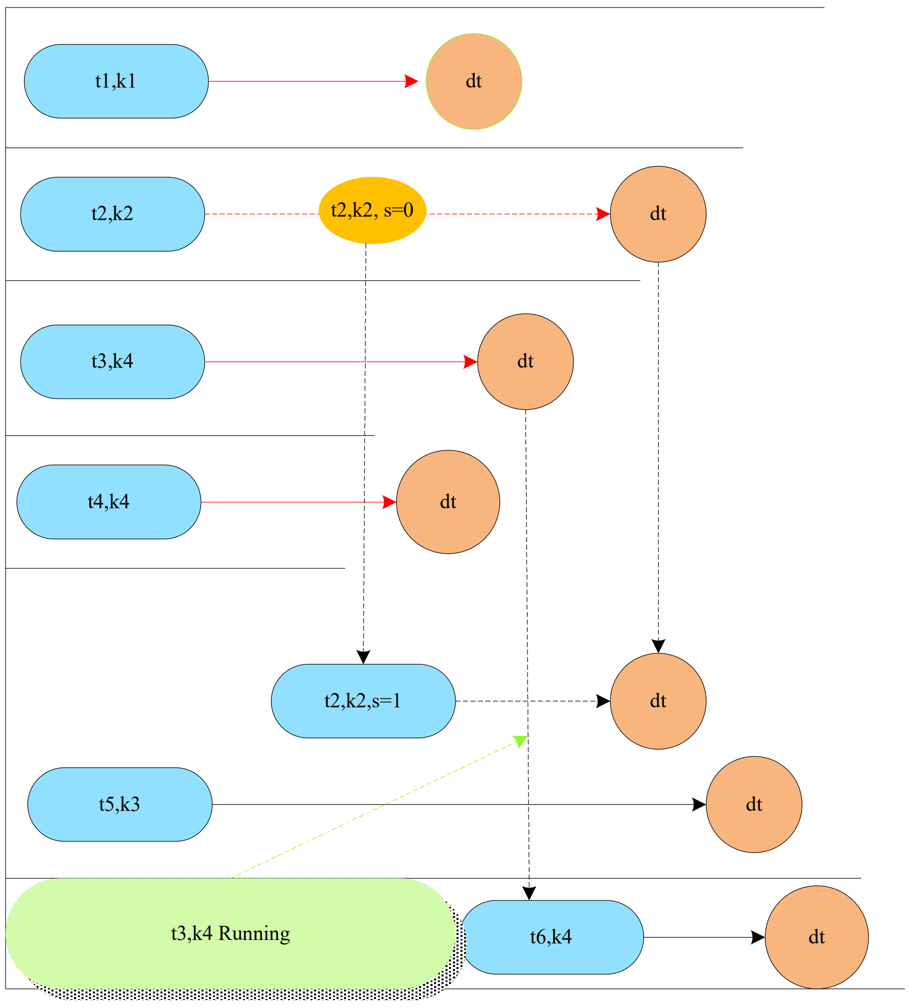


Figure 3. Scheduling under deadlines.

under its deadline on the same or another node as shown in Figure 3. The failure task can be scheduled on the same node or reschedule another node from the point of failure. In this way, the scheduler can dynamically handle the tasks' execution and failure in the heterogeneous fog networks. In our problem, all edge nodes are homogenous. Therefore, scheduling from one node to another

Algorithm 4: Fault-Tolerant Delay Scheme

```
  Input  : $(List - f[t, k, s])$;
1 begin
2     foreach $(List - f[t, k, s])$ do
3         Apply primary backup and checkpointing strategies;
4         Reschedule $t3 \leftarrow k1$ to $t3 \leftarrow k2$;
5         $[t3, k2, s] = 1$;
6     Repeat until failure tasks with status 0 become 1;
7 End loop;
```

based on available resources is convenient in our model. We implemented two well-known techniques of fault-tolerant such as primary backup [27] and checkpointing [28] in edge nodes.

Figure 3 shows the scheduling process of tasks on to the heterogeneous fog nodes under the deadline and assignment constraints. The blue node shows that a task t is assigned to the fog node k for the execution. All the tasks are assigned based on the deadline of tasks on to the available nodes. The purple color shows that the assigned tasks on nodes met their deadlines during scheduling in the system. If two jobs (e.g., tasks) have arrived at the same node, they will schedule based on their deadlines. Furthermore, the next task will wait until the completion of the first task execution. For instance, $t3, k4$ is already running as shown in the green-covered area, and then task $t6, k4$ will wait until finish time $t6$ during scheduling. If the task failed as shown in Figure 3 with yellow node, and its status changed from status 1 to 0. The scheduler will call Algorithm 4 to recover tasks with the checkpointing technique on the primary backup scheme in the network. For instance, a task $t2$ failed on node $k2$, but it has a deadline. It recovers under its deadline on the same or another node as shown in Figure 3. The failure task can be scheduled on the same node or reschedule another node from the point of failure. In this way, the scheduler can dynamically handle the tasks' execution and failure in the heterogeneous fog networks.

5. Performance evaluation and result discussion

This section manifests the performance evaluation, system implementation, baseline approaches, and results from the discussion from work. Initially, the study designed the RPC system as shown in Figure 4 based on different parameters and values. For instance, the parameters discussed and demonstrated in Table 3 are considered in the simulation environment where other types of resources are deemed as shown in Table 4. The study designed the new RPC environment as shown in Figure 4 for IIoHT applications based on the edge foundry platform for simulation purposes. The study created a practical RPC system consisting of an add services module. Many healthcare services can install on Android devices that are deployed at fog nodes in different hospitals. The services are healthcare tasks with various tasks contents such as text, video, images, and audio, and these tasks are encrypted and decrypted at the client device and fog node with the public key and private key. The RPC consists of two sockets classes, such as client-socket and server socket. Whereas add services and encryption and decryption modules connected and write and read components in the system. The server socket class has different elements such as connection bind, accept and listen, which ensure the connection between the local device socket and server socket for further processing. The data offload in Javascript Object Notation (JSON) between client socket and server socket in the system. The server socket implemented the edge foundry SDK library in RPC. The edge foundry is an open Industrial Internet of Healthcare Things (IIoHT) edge cloud platform which can create fog nodes based on the container's docker images at the repository. The container is the resources on the fog node and processes the tasks based on their size in the inside socket and managed by the docker Kubernetes manager. The framework is a modified version of RPC and edge foundry tested at Android devices and edge cloud networks for all IIoHT bio-sensors applications. The RPC is designed based on JAVA programming, which can offload and run applications at different layers such as clients and servers with the same runtime Java virtual machine environment. All the bio-sensors are healthcare and can connect with various sensors; we only generate sensors' data in the proposed RPC system in work. Table 4 shows the implemented fog nodes in the

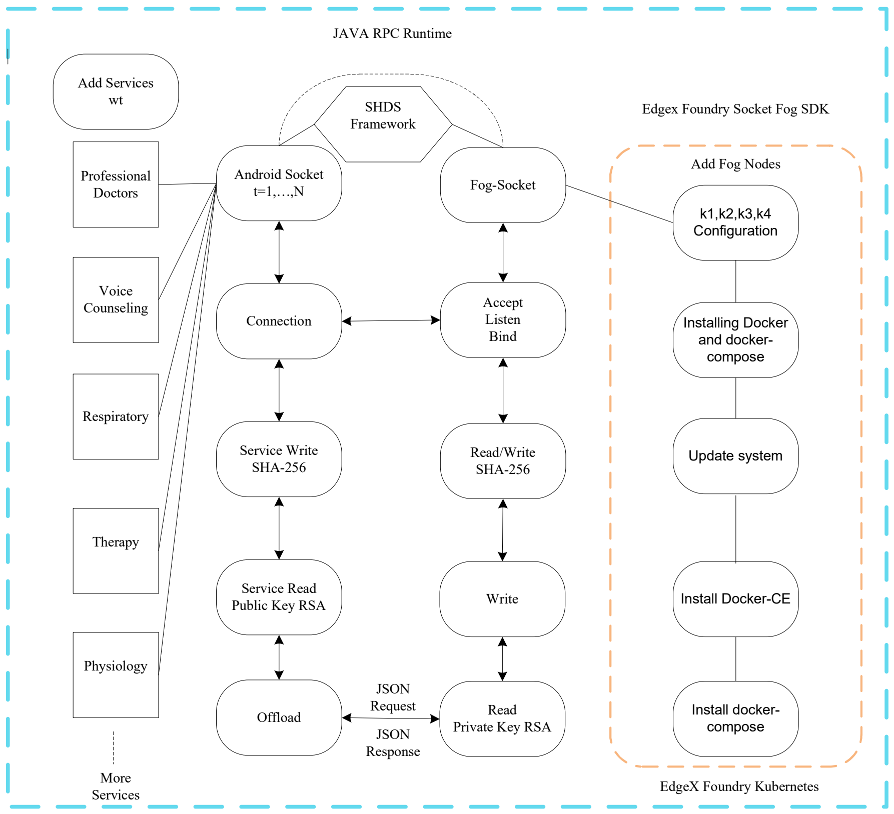


Figure 4. Implementation of hybrid delay optimal bio-sensors connected RPC enabled edge computing IIoHT system.

Table 3. Simulation RPC enabled edge computing IIoHT system parameters.

Parameters	Values
Programming	Socket
N	1000
$\sum_{t=1}^{N} W_t$	1000GB
K	4
k1	i5th Core
k2	i7th Core
k3	i9th Core
k4	Highth Core

Table 4. EdgeX foundry fog nodes.

N	ϵ_k	ζ_kMPS
k_1	1000GB	Dual Core i5
k_2	5000GB	Dual Core i7
k_3	10000GB	Dual Core i9
k_4	20000GB	Dual Core High

edgex foundry SDK. Each fog node is heterogeneous and has different storage and processing speed in the system. The higher fog node has less processing delay as compared to low processing fog node, therefore, due to a lot of traffic high processing fog nodes are not available for the execution of requests in the system. The study managed the workload among different fog nodes under their deadline and delay.

5.1. Baseline approaches result discussion

The study implements the existing RPC schemes as the baseline approaches in the simulation environment. These studies [3, 6, 8, 19, 21, 22] have been assumed the baseline (B1) delay optimal suggested schemes in which only network delay and computational delay considered in the RPC for the healthcare applications. Furthermore, these studies [2, 4, 7, 15, 20, 26] have been assumed the baseline (B2) delay optimal suggested schemes in which only failure delay and computational delay considered in the RPC for the healthcare applications. These studies are closely related to our work, and they assumed jobs to the healthcare tasks in the RPC system during the simulation environment in their works. Therefore, we considered the tasks as the jobs of IIoHT applications in the simulation environment with the given parameters.

5.1.1. Security delay of IIoHT applications

In the existing studies, the security delay is widely ignored for the applications. At the same time, the security delay is time-consuming and requires a lot of resources for encrypting the healthcare data. The study devises the lightweight homomorphic SHA-256 encryption and decryption schemes based on the asymmetric mechanism for the applications. Initially, the offloader engine checks if the local devices, e.g., Android nodes, have enough resources to encrypt and decrypt data on the healthcare workload; it calls SHA-256 security schemes and encrypts workload based on the given key, which is generated based on RSA [4]. However, if the local devices have limited resources and can't do encryption offloader engine offload all data to the fog node in RPC for the processing in the system. It means the proposed security method in SHDS is more dynamic and efficient, making decisions at the runtime to avoid delay in the applications. Figure 5 shows the Z performances of all tasks (e.g., jobs) with the baseline approaches such as existing secure offloading schemes as compared to the SHDS proposed method. Figure 5 shows that SHDS outperformed all existing secure offloading procedures in terms of objective Z, which consists of network delay and computation delay. During the simulation, 600 and 800 tasks are encrypted and decrypted, and their security delay time is calculated at different nodes in the simulated RPC system. It is assumed from the obtained results in Figure 5, the existing studies failed to optimize the security, which has a direct impact on the objective function of the healthcare applications. Figure 6 shows the objective function performances of IIoHT applications with different numbers of jobs that have a direct impact due to security delay. It is, therefore, is very important, the security delay is calculated in RPC for IIoHT applications. Otherwise, it is hard to satisfy the security and deadline requirements of applications together in RPC. In the security mechanism, we have chosen the two security offloading schemes with the MD5-256 bits and SHA-256 in RPC for healthcare applications. These studies make encryption and decryption

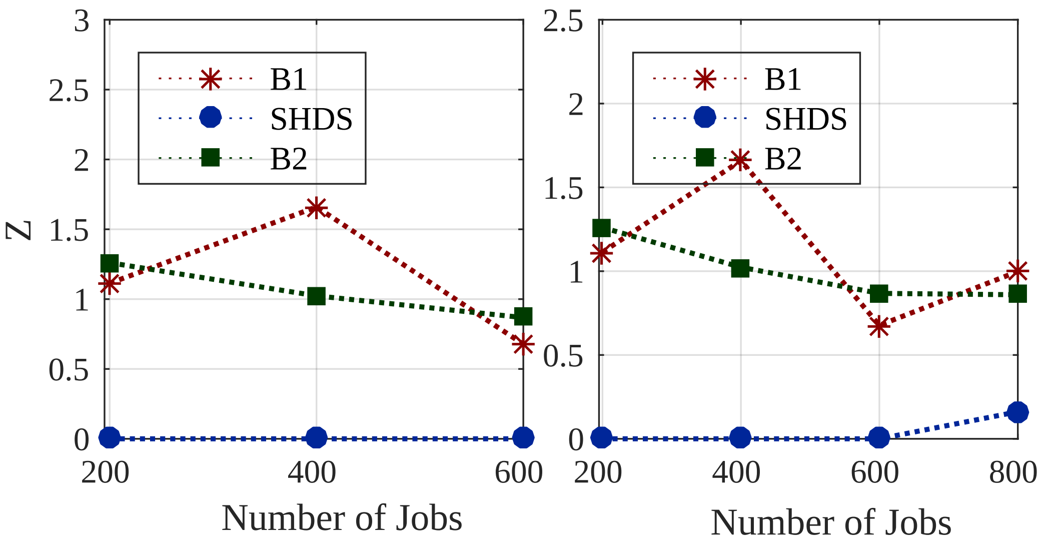
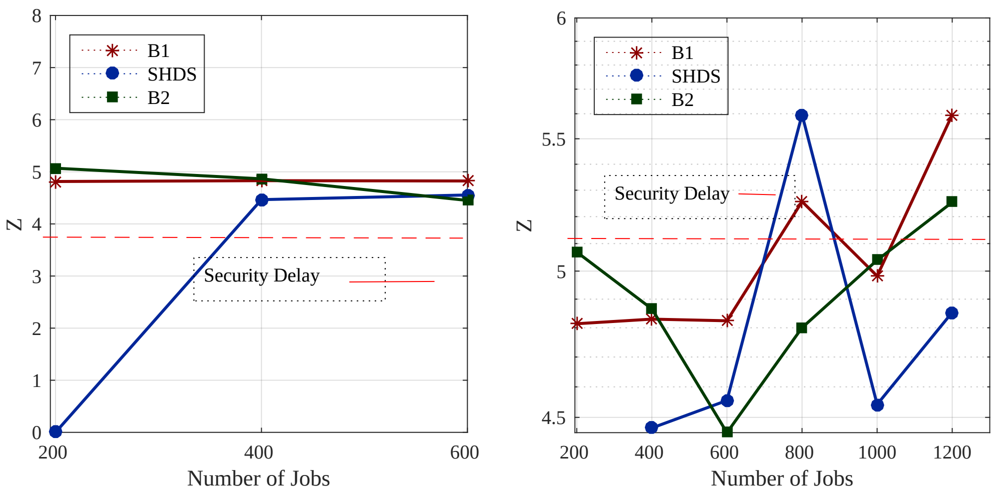


Figure 5. Performances of security delay of IIoHT applications at different nodes.



(a) Number of 850 Jobs **(b)** Number of 1000 Jobs

Figure 6. Optimal performance of secure offloading schemes B1 (SHA-256) and B2 (MD5) for IIoHT applications.

locally on mobile due to security validation. However, due to resource constraints of mobile devices, sometimes encryption and decryption cannot be performed locally in the system. Due to this reason, many failures occurred at the local machines. Therefore, the offloader engine in our work decides the runtime offloading decision where to encrypt and decrypt tasks data based on available resources in different nodes. In this way, we can optimize the security constraint delay for IIoHT applications with varying numbers of tasks (850 and 1000) in the RPC. Figure 6 (a) and (b) showed the objective function with the SHDS is more optimal than existing schemes where decisions made at the design time and occur at the devices and incurred with the failure and a lot of processing delay in the RPC.

The network delay directly impacts the objective function when it offloads and downloads healthcare

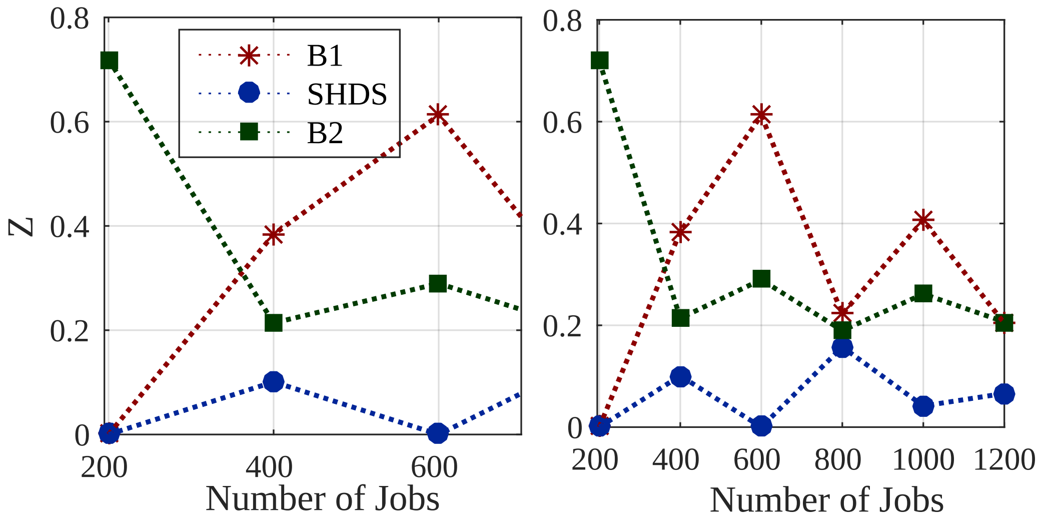
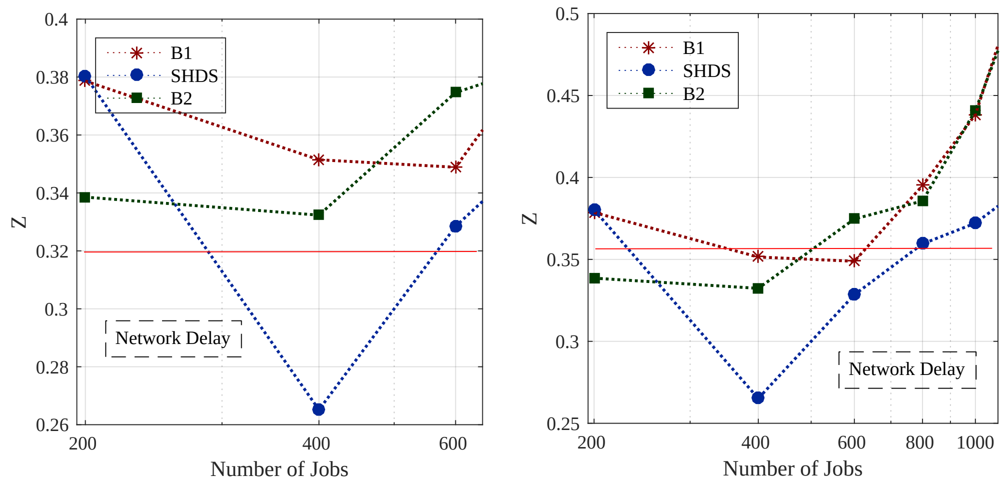


Figure 7. Network delay optimal offloading over objective function for IIoHT applications.

data for the processing and results. Generally, wireless technologies have a different offloading and downloading delay on IIoHT data in RPC. However, we are comparing only schemes of network delay implemented in RPC. Figure 7 shows the objective function performances of network delay with the different number of tasks (600 and 1200) based on existing baseline approaches in RPC. Figure 7 shows the SHDS outperformed all existing schemes in terms of network delay in RPC for the IIoHT applications. The main reason is that SHDS determined the network delay before offloading the encrypted tasks to the system based on the given deadline. If the network delay has a longer delay and tasks still miss their deadline based on obtained network delay, the offloader engine in the study makes the tasks fail and resubmit from scratch in the RPC. However, in the proposed RPC, the user



(a) Number of 1000 Number of Jobs

(b) Number of 1200 1000 Number of Jobs

Figure 8. Network delay of IIoHT applications based on their deadlines.

still offloads their tasks without losing their deadlines in the worst case of the network in RPC. However, existing network delay offloading schemes only offloaded tasks without considering the deadline of tasks of applications in RPC. These approaches have much more failure or an extended network delay with the possibility that many more jobs missed their deadlines. Therefore, SHDS is more efficient and optimal than existing offloading network delay schemes in RPC in terms of the objective function in the system as shown in Figure 8 (a),(b). The computational delay belongs to the

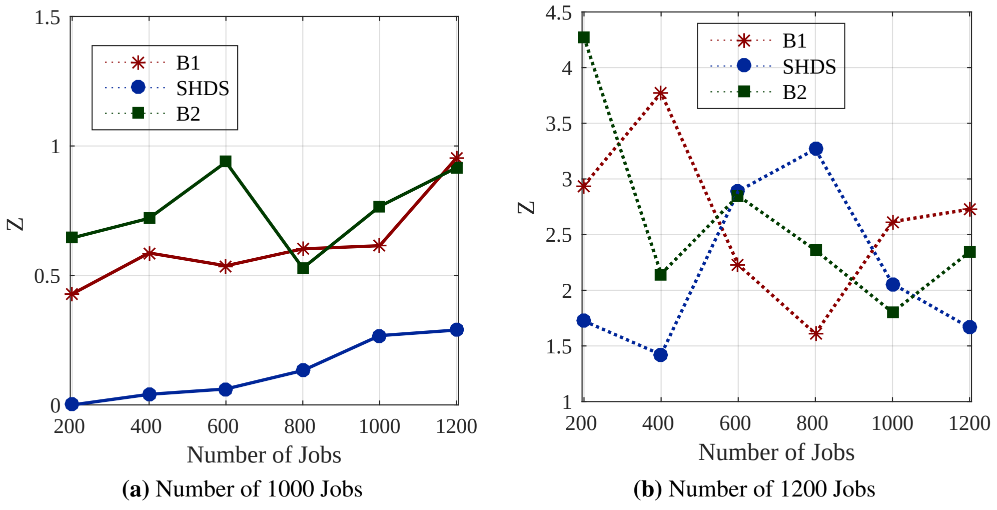


Figure 9. Computational delay of IIoHT applications based on their deadlines.

scheduling of tasks to the available nodes based on resources in the RPC. Most of the existing RPC exploited the virtual machine fog nodes with different speeds and storage to run IIoHT healthcare tasks under their deadlines. However, many factors can degrade the performances of computational delay in RPC, such as resource boot-up time, wait time, and execution time. Therefore, lightweight boot-up-time and stay and execution must be optimal for IIoHT applications in RPC based on the mobile edge cloud system. In the simulation environment, we executed 1200 healthcare jobs in the RPC and analyzed their performances in the RPC based on their deadlines. Figure 9 (a),(b) shows that, with different boot-up-time, wait time and execution has a direct impact on the objective function of the healthcare application with the different number of tasks. Figure 9 (a),(b) shows that the objective function is improved with the proposed RPC scheme as compared to baseline RPC schemes in terms of computational delay for the IIoHT applications. The main reason is that the study modified and improved the performance of RPC with the boot-up-time delay and wait for delay execution delay compared to the existing RPC. The main reason is that the current studies implemented virtual machines with 55 boot-up-time and longer wait times when executing tasks on the devices. However, the proposed schemes implemented docker containers in edgex foundry with only 11 seconds of boot-up time and zero wait time in the RPC. Why boot-up-time is more important in the computational delay, the main reason is that the resource cannot always be in ”ON” status because in ”ON” situation consumes much more resources in the system. Therefore, they need to restart and incur with higher boot-up time in the RPC. Therefore, existing studies ignored the boot-up-time delay and waited for the delay in RPC for the IIoHT applications.

Besides security delay, network delay, offloading delay, and fault-tolerant delay are most significant

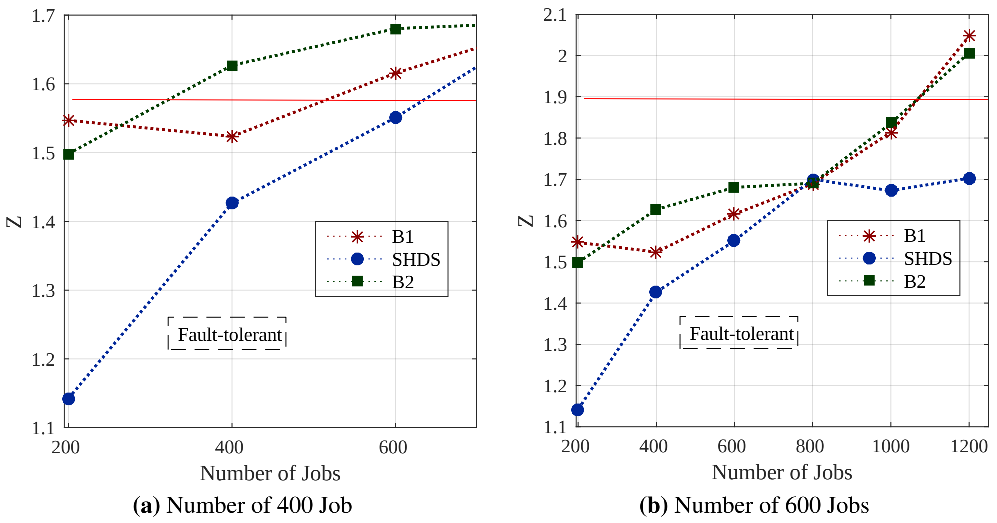


Figure 10. Fault delay of IIoHT applications based on their deadlines.

in RPC when it offers the seamless and robust JAVA runtime for IIoHT applications. The existing fault-tolerant techniques such as checkpointing and primary backup are widely implemented in the RPC with the proposed offloading and resource-allocations schemes. Figure 10 (a),(b) shows that checkpointing and primary backup always incurred with additional delay and still meet the deadline requirements with the proposed scheme SHDS. However, existing schemes widely ignored the fault-tolerant delay in RPC for the IIoHT applications.

5.2. *Research finding and limitations*

The main finding of this study devised the lightweight RPC runtime framework for IIoHT applications where many types of delay are considered the constraints in the system. The delays are security, network, computational, and fault-tolerant delays for IIoHT applications under their deadlines. Whereas existing studies only considered the computational delay and network constraints for IIoHT applications in RPC systems. Therefore, this RPC framework still missed a huge of applications deadlines. The proposed SHDS schemes and lightweight RPC optimized the objective function of IIoHT applications with the hybrid delays, as shown in the performances in the result discussion part. However, this study has limitations that are to be investigated further in future work of the study. For instance, mobility delay, migrations delay, and many delays are still to be investigated further in the current model of RPC for the IIoHT applications.

6. Conclusions and future work

In this study, the work devised the lightweight RPC mechanism for IIoHT applications and considered the hybrid constraints in the system. The study suggests the secure hybrid delay scheme (SHDS), which schedules all healthcare workloads under their deadlines. For the scheduling problem, the study formulated this problem based on linear integer programming, where all constraints are integer, as shown in the mathematical model. The results-discussion showed that the proposed RPC and its scheme SHDS minimized hybrid delays such as security delay, network delay, computational delay, and fault-tolerant delay compared to existing RPC schemes. The study showed that the hybrid delay performances of IIoHT applications could be minimized by 50% and proved in the result discussion part. The study will consider the mobility delay and roaming delay of applications in the current version of RPC in the future. RPC has many room delays that can be improved, such as wait delays, migration delays, services delays, etc. Therefore, we will solve the IIoHT combinatorial problem with both concave and convex functions with many convex sets in the RPC system in our future work.

Acknowledgments

This work is financially supported by the Research grant of PIFI 2020 (2020VBC0002), Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences (SIAT,CAS), Shenzhen, China.

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. L. A. Mastoi, Q. Mastoi, M. Elhoseny, M. S. Memon, M. A. Mohammed, Deep neural network-based application partitioning and scheduling for hospitals and medical enterprises using iot assisted mobile fog cloud, *Enterp. Inf. Syst.*, (2021), 1–23. doi: 10.1080/17517575.2021.1883122.
2. H. Zhu, P. Tiwari, A. Ghoneim, M. S. Hossain, A collaborative ai-enabled pretrained language model for aiot domain question answering, *IEEE Trans. Ind. Inf.*, (2021). doi: 10.1109/TII.2021.3097183.
3. A. Lakhan, M. Ahmad, M. Bilal, A. Jolfaei, R. M. Mehmood, Mobility aware blockchain enabled offloading and scheduling in vehicular fog cloud computing, *IEEE Trans. Intell. Transp. Syst.*, (2021), doi: 10.1109/TITS.2021.3056461.
4. S. Mishra, H. Thakkar, P. K. Mallick, P. Tiwari, A. Alamri, A sustainable ioht based computationally intelligent healthcare monitoring system for lung cancer risk detection, *Sustainable Cities Soc.*, 103079, (2021). doi: 10.1016/j.scs.2021.103079.
5. A. Lakhan, M. S. Memon, M. Elhoseny, M. A. Mohammed, M. Qabulio, M. Abdel-Basset, et al., Cost-efficient mobility offloading and task scheduling for microservices iovt applications in container-based fog cloud network, *Cluster Comput.*, (2021), 1–23. doi: 10.1007/s10586-021-03333-0.
6. A. Lakhan, M. A. Mohammed, A. N. Rashid, S. Kadry, T. Panityakul, K. H. Abdulkareem, et al., Smart-contract aware ethereum and client-fog-cloud healthcare system, *Sensors*, **21** (2021), 4093. doi: 10.3390/s21124093.

7. A. Lakhan, M. A. Dootio, T. M. Groenli, A. H. Sodhro, M. S. Khokhar, Multi-layer latency aware workload assignment of e-transport iot applications in mobile sensors cloudlet cloud networks, *Electronics*, **10** (2021), 1719. doi: 10.3390/electronics10141719.

8. M. Hussain, L. F. Wei, A. Lakhan, S. Wali, S. Ali, A. Hussain, Energy and performance-efficient task scheduling in heterogeneous virtualized cloud computing, *Sustainable Comput. Inf. Syst.*, **30** (2021), 100517. doi: 10.1016/j.suscom.2021.100517.

9. A. Lakhan, X. Li, Transient fault aware application partitioning computational offloading algorithm in microservices based mobile cloudlet networks, *Computing*, **102** (2020), 105–139. doi: 10.1007/s00607-019-00733-4.

10. A. Lakhan, L. Xiaoping, Energy aware dynamic workflow application partitioning and task scheduling in heterogeneous mobile cloud network, in *2018 International Conference on Cloud Computing, Big Data and Blockchain (ICCBB)*, IEEE, (2018), 1–8. doi: 10.1109/ICCBB.2018.8756442.

11. A. Lakhan, X. Li, Content aware task scheduling framework for mobile workflow applications in heterogeneous mobile-edge-cloud paradigms: Catsa framework, in *2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, IEEE, (2019), 242–249. doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00044.

12. A. Lakhan, X. Li, Mobility and fault aware adaptive task offloading in heterogeneous mobile cloud environments, *EAI Endorsed Trans. Mobile Commun. Appl.*, **5** (2019). doi: 10.4108/eai.3-9-2019.159947.

13. J. Qian, P. Tiwari, S. P. Gochhayat, H. M. Pandey, A noble double-dictionary-based ecg compression technique for ioth, *IEEE Internet Things J.*, **7** (2020), 10160–10170. doi: 10.1109/JIOT.2020.2974678.

14. F. Zhang, M. M. Wang, Stochastic congestion game for load balancing in mobile edge computing, *IEEE Internet Things J.*, (2020). doi: 10.1109/JIOT.2020.3008009.

15. A. Lakhan, Q. Mastoi, M. A. Dootio, F. Alqahtani, I. R. Alzahrani, F. Baothman, et al., Hybrid workload enabled and secure healthcare monitoring sensing framework in distributed fog-cloud network, *electronics*, **10** (2021), 1974. doi: 10.3390/electronics10161974.

16. F. H. Khoso, A. Lakhan, A. A. Arain, M. A. Soomro, S. Z. Nizamani, K. Kanwar, A microservice-based system for industrial internet of things in fog-cloud assisted network, *Eng., Technol. Appl. Sci. Res.*, **11** (2021), 7029–7032. doi: 10.48084/etasr.4077.

17. F. H. Khoso, A. A. Arain, A. Lakhan, A. Kehar, S. Z. Nizamani, Proposing a novel iot framework by identifying security and privacy issues in fog cloud services network, *Int. J*, **9** (2021), 592–596. doi: 10.30534/ijeter/2021/10952021.

18. A. Lakhan, R. Singh, Implementation of etl tool for data warehousing for non-hodgkin lymphoma (nhl) cancer in public sector, pakistan, *Int. J.*, **9** (2021). doi: 10.30534/ijeter/2021/27972021.

19. A. Lakhan, F. H. Khoso, A. A. Arain, K. Kanwar, Serverless based functions aware framework for healthcare application, *Int. J.*, **9** (2021). doi: 10.30534/ijeter/2021/19942021.

20. U. Rehman, M. A. S. A. Lakhan, A review on state of the art in flipped classroom technology a blended e-learning, *Int. J.*, **9** (2021). doi: 10.30534/ijeter/2021/22972021.

21. I. A. Jamali, A. Lakhan, D. Kumar, A. R. Mahessar, R. lodhi, Energy efficient task assignment algorithm framework in mo-bile cloud computing, *GSJ*, **6** (2018), 171.

22. A. L. Mujeeb-ur Rehman, Z. Hussain, F. H. Khoso, A. A. Arain, Cyber security intelligence and ethereum blockchain technology for e-commerce, *Int. J.*, **9** (2021).

23. A. Lakhan, D. K. Sajnani, M. Tahir, M. Aamir, R. Lodhi, Delay sensitive application partitioning and task scheduling in mobile edge cloud prototyping, in *International Conference on 5G for Ubiquitous Connectivity*, Springer, (2018), 59–80.

24. D. K. Sajnani, A. R. Mahesar, A. Lakhan, I. A. Jamali, R. Lodhi, M. Aamir, Latency aware optimal workload assignment in mobile edge cloud offloading network, in *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, IEEE, (2018), 658–662. doi: 10.1109/CompComm.2018.8780954.

25. D. K. Sajnani, A. R. Mahesar, A. Lakhan, I. A. Jamali, et al., Latency aware and service delay with task scheduling in mobile edge computing, *Commun. Network*, **10** (2018), 127. doi: 10.4236/cn.2018.104011.

26. A. H. Sodhro, Z. Luo, A. K. Sangaiah, S. W. Baik, Mobile edge computing based qos optimization in medical healthcare applications, *Int. J. Inf. Manage.*, **45** (2019), 308–318. doi: 10.4236/cn.2018.104011.

27. A. H. Sodhro, S. Pirbhulal, V. H. C. De Albuquerque, Artificial intelligence-driven mechanism for edge computing-based industrial applications, *IEEE Trans. Ind. Inf.*, **15** (2019), 4235–4243.

28. M. Muzammal, R. Talat, A. H. Sodhro, S. Pirbhulal, A multi-sensor data fusion enabled ensemble approach for medical data from body sensor networks, *Inf. Fusion*, **53** (2020), 155–164. doi: 10.1109/TII.2019.2902878.

29. H. Magsi, A. H. Sodhro, F. A. Chachar, S. A. K. Abro, G. H. Sodhro, S. Pirbhulal, Evolution of 5g in internet of medical things, in *2018 international conference on computing, mathematics and engineering technologies (iCoMET)*, IEEE, (2018), 1–7. doi: 10.1109/ICOMET.2018.8346428.

30. T. Zhang, A. H. Sodhro, Z. Luo, N. Zahid, M. W. Nawaz, S. Pirbhulal, et al., A joint deep learning and internet of medical things driven framework for elderly patients, *IEEE Access*, **8** (2020), 75822–75832. doi: 10.1109/ACCESS.2020.2989143.

31. T. Li, Z. Wang, Y. Chen, C. Li, Y. Jia, Y. Yang, Is semi-selfish mining available without being detected? *Int. J. Intell. Syst.*, 2021. doi: 10.1002/int.22656.

32. A. A. Mutlag, M. K. A. Ghani, M. A. Mohammed, A. Lakhan, O. Mohd, K. H. Abdulkareem, et al., Multi-agent systems in fog-cloud computing for critical healthcare task management model (chtm) used for ecg monitoring, *Sensors*, **21** (2021), 6923. doi: 10.3390/s21206923.

33. X. Yu, Z. Wang, Y. Wang, F. Li, T. Li, Y. Chen,et al., Impsuic: A quality updating rule in mixing coins with maximum utilities, *Int. J. Intell. Syst.*, **36** (2020), 1182–1198.

34. T. Li, Y. Chen, Y. Wang, Y. Wang, M. Zhao, H. Zhu, et al., Rational protocols and attacks in blockchain system, *Secur. Commun. Networks*, **2020** (2020), 1–11. doi: 10.1155/2020/8839047.

35. G. Yang, Y. Wang, Z. Wang, Y. Tian, X. Yu, S. Li, Ipbsm: An optimal bribery selfish mining in the presence of intelligent and pure attackers, *Int. J. Intell. Syst.*, **35** (2020), 1735–1748. doi: 10.1002/int.22270.

36. Y. Wang, G. Yang, T. Li, L. Zhang, Y. Wang, L. Ke, et al., Optimal mixed block withholding attacks based on reinforcement learning, *Int. J. Intell. Syst.*, **35** (2020), 2032–2048. doi: 10.1002/int.22282.

37. X. Liu, X. Yu, H. Zhu, G. Yang, Y. Wang, X. Yu, et al., A game-theoretic approach of mixing different qualities of coins, *Int. J. Intell. Syst.*, **35** (2020), 1899–1911. doi: 10.1002/int.22277.

38. Ö. Çelikel, T. Ovatman, Distributed application checkpointing for replicated state machines, *Scalable Comput.: Pract. Exper.*, **22** (2021), 67–79. doi: 10.12694/scpe.v22i1.1840.

39. R. Wang, N. Chen, X. Yao, L. Hu, Fasdq: Fault-tolerant adaptive scheduling with dynamic qos-awareness in edge containers for delay-sensitive tasks, *Sensors*, **21** (2021), 2973. doi: 10.3390/s21092973.