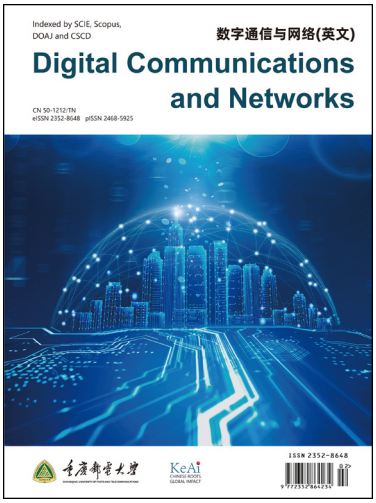
Journal Pre-proof



A latency sensitive and agile IIoT architecture with optimized edge node selection and task scheduling

Mona Kumari, Maheswari Prasad Singh and Amit Kumar Singh

PII: S2352-8648(25)00048-3
DOI: https://doi.org/10.1016/j.dcan.2025.04.012
Reference: DCAN 869

To appear in: *Digital Communications and Networks*

Received date: 25 September 2024
Revised date: 16 April 2025
Accepted date: 28 April 2025

Please cite this article as: M. Kumari, M.P. Singh and A.K. Singh, A latency sensitive and agile IIoT architecture with optimized edge node selection and task scheduling, *Digital Communications and Networks*, doi: https://doi.org/10.1016/j.dcan.2025.04.012.

Available online at www.sciencedirect.com

ScienceDirect

00 (2025) 1–13

Journal Logo

www.elsevier.com/locate/procedia

A latency sensitive and agile IIoT architecture with optimized edge node selection and task scheduling

Mona Kumari[a], Maheswari Prasad Singh*[a], Amit Kumar Singh[a]

[a]*Department of Computer Science and Engineering, NIT Patna,Bihar, 800005, India*

Abstract

The Industrial Internet of Things (IIoT) has revolutionized conventional manufacturing industries by improving control, monitoring, and management, resulting in increased agility and long-term sustainability.The demand for processing and storage resources increases as the number of IoT devices and the data they generate increases. Cloud computing often serves these needs, but in certain scenarios, real-time data processing near the source is necessary to guarantee low latency, minimize bandwidth usage, and enhance data security. This work proposes a novel edge computing-based IIoT architecture for delay-sensitive monitoring and control in manufacturing industries.In contrast to previous work, this work is designed to enhance system performance by ensuring the availability of edge nodes and distributing task execution loads evenly across multiple edge computing nodes. The proposed architecture is further optimized for efficient allocation of resources and scheduling of tasks by the addition of a task scheduling algorithm by using Round Trip Time (RTT) and Resource Logger (RL). In an effort to optimize the response time of both the complete system and individual Device Nodes (DN), the work also considers the deployment of heterogeneous Edge Nodes (EN). The evaluation results show that the proposed architecture reduces the communication delay and overhead by 88.28% compared to the traditional cloud-based approach and 12.32% compared to the previously proposed methods ATS-FOA and RBEC/PENB, respectively, for executing IIoT applications. The proposed Resource Logger, Round Trip Time-Based Edge Node Selection (RLRTT-ESA), and task scheduling algorithms further improved the total application response time by 47.25% in comparison to the previously proposed methods ATS-FOA and RBEC/PENB, respectively.

Keywords:
IoT, IIoT, Edge computing, Delay sensitive, RTT, Load balancing, RL

1. Introduction

Today, one of the most significant technologies scaling new heights across industry segments is the Internet of Things (IoT). IoT is defined as an approach that interconnects real-world objects, sensors, actuators, machines, computers etc. to the Internet for meaningful purposes. The IoT has expanded into various aspects of life, such as consumer products, utility components, industries, healthcare, the supply chain, and everyday objects, with the potential to revolutionize our way of living and working [1, 2].Examples of IoT applications can be as simple as turning on a light bulb using a smartphone or a motion sensor, or they can extend to larger deployments like a manufacturing shop floor, where thousands of sensors, robotics, and actuators are deployed to collect, process, and transmit results, ensuring efficient operations [3]. The rapid development of IoT applications and increase in the number and variety of these devices necessitate the intelligent storage and processing of a vast volume of data to extract valuable insights. Traditionally, this large amount of data collected needs to be transferred to the cloud for processing [4].This poses a challenge due to network issues such as data transfer protocol, data loss, security, compliance, and various other factors [5].This is where IoT edge computing systems are introduced to eliminate the need for transmitting and processing every piece of data in the cloud [6]. Edge computing, a decentralized computing infrastructure, distributes computing systems and application services along a communication path close to the data source. These systems, provisioned at the edge, meet computational needs by collecting, storing, and processing data from the source [7].

Smart manufacturing industries based on the IIoT are becoming increasingly popular. These industries are made feasible by the Internet of Things (IoT) and wireless communication technology [8].In these industries, manufacturing machines and equipment's remain interconnected and communicate with each other to coordinate its activities, and multiple sensors are deployed with or between machines to enhance the visibility of processes, equipment's, and the status of products for better monitoring and control of the complete industrial system from anywhere.

Manufacturing is one of the industries most impacted and progressed by IoT. This has drastically changed the way most of the manufacturing process is carried out, from the supply chain to the delivery of finished goods for distribution, and it has been called the fourth industrial revolution, or Industry 4.0. The IIoT is transforming traditional, lin-

*Corresponding author
[1]Email address:monak.ph21.cs@nitp.ac.in, mps@nitp.ac.in,amit.singh@nitp.ac.in

ear manufacturing into dynamic, interconnected systems. It enhances efficiency by automating and optimizing each stage of the production process. In addition, the monitoring of daily operations, the health of machinery, the reduction of energy consumption, and carrying out preventive maintenance. These IoT devices gather data that gives real-time insights and actionable information, which facilitates better decision-making and guarantees timely notifications.

The use of IoT in managing manufacturing industrial processes [9] which necessitates the addition of numerous sensors to the industrial environment for data collection. These data undergo initial and advanced processing, including removing duplicates, arranging data in the required format, context-specific data segregation, and some complex processing that is based on artificial intelligence and machine learning models [10]. The collected data can be effectively mined for valuable information through the use of all of these processes. We also use various data visualization models to present information in an effective and summarized form, facilitating easy and quick user perception. All these data processing and visualization models require a numerous computing and storage resources [11].

Cloud computing is an approach that addresses the limitations of associated devices in the IIoT in terms of storage, networking, complex computing, analysis, scalability, and continuous access to a large pool of data from anywhere, anytime, by anyone. In this scenario, even in a modest industry, the amount of data generated could overwhelm the existing infrastructure and ramp up the significant cost of bandwidth, storage, and computing systems. Although cloud computing and storage are necessary for the IIoT, the transmission between data generation, processing, and return can result in issues with data security, data management, bandwidth, and latency. Edge computing mitigates this issue by assuring that data processing occurs at the outer edges, close to the data source. This results in a predictable and ultra-low latency that is optimal for time-critical situations [12], where the operation is either mission-critical or where objects are in motion. Additionally, it increases the speed of data processing. The data is guaranteed to be legible, accessible, secure, and compliant by edge computing devices [13].

Despite the fact that cloud computing can be useful in overcoming certain IIoT limitations, there are situations in which millions of devices continuously send large amount of useless data to the cloud. Additionally, low latency, real-time, location awareness, geo-distribution, and mobility support must be addressed, and cloud computing is unable to address these challenges [14]. Storage, computation, and networking services will be provided between the cloud and end devices by a new platform known as edge computing. The primary objective is to expand computation in order to facilitate its integration with IIoT devices. Edge computing is a distributed and collaborative platform that, in conjunction with IIoT devices, offers essential services [15].

Edge computing may be defined as the deployment of data aggregation and processing power close to the network edge of the IIoT devices that produce data. For IIoT deployment, the edge will be the computing resources such as industrial machinery or equipment, gateways, protocol converters, or other types of industrial control panels that analyze the data and identify and transmit only the relevant data back to the on-site data center and cloud for further processing [16]. The recent development of edge computing provided an idea for the decentralization of industrial automation systems. One of the most prominent solutions for the implementation of IoT architectures that incorporate industrial automation and real-time control is edge computing [17]. The volume of data transfer in the IIoT is significantly increasing. The high frequency of data transmissions is one factor contributing to this. In a real-time application, an IIoT device normally transmits a data load every few seconds. Additionally, IIoT applications necessitate significantly more complex and resource-intensive data processing. This is due to the fact that IIoT applications generally require the coordination of numerous devices.

By decentralizing and distributing computing resources across the edge and far less data is flowing through the network, which frees up costly bandwidth for other applications, saves processing time, improves data processing speed and data accessibility and visibility [18].Cloud computing and edge computing are complementary in that an analytical model or rules are developed in the cloud and subsequently distributed to the edge devices.

This work examines the deployment of multiple heterogeneous edge computing nodes as local servers in an industrial manufacturing environment in order to improve the response time of the entire system and individual industrial IoT devices, for end-to-end real-time process visibility and availability of computing and storage resources at the edge of the industry to recover from single point of failure. In addition, it also offers a task scheduling technique to address the load fluctuation that is caused by the dynamic characteristics of user movements, mobile devices, and equipment in an industrial environment. The main contributions of this work are summarized as follows:

1. Edge-cloud computing IIoT based system architecture, consisting of multiple mobile and static industrial assets, machines and control panels as IIoT device or Device Node(DN) equipped with the data collection and communication techniques, n Device Coordinator (DC), m Edge Node (EN), Resource Logger (RL) and centralized cloud server.
2. Manage congestion and uniformly distributing the task execution load on each individual EN. A task scheduling algorithm is proposed in this work for resource allocation and computation load forwarding, from DC_n to EN_m such that it can get processed with minimum delay.

The rest of the work in this paper is organized as follows. **Section 2** presents related work. **Section 3** presents the problem definition and an overview of proposed solution. **Section 4** presents the proposed system architecture and response time model. **Section 5** presents solution approaches and algorithm in which task scheduling model and initial solution based on approximate nearest edge node selection solved by Locality Sensitive Hashing is also define. Simulation setup, device details, result and insight of proposed solution is presented in **Section 6**. Finally, **Section 7** concludes the work.

2. Related Work

One intriguing paradigm that can enable Industrial Internet of Things (IIoT) applications that are delay-sensitive is the convergence of edge and cloud computing. This study examines existing research on edge-cloud computing systems and their use in industrial environments. However, future research and development efforts will continue to be heavily focused on overcoming technical issues and ensuring effortless integration. In an effort to capitalize on the advantages of both paradigms, recent efforts have concentrated on merging edge and cloud computing. A novel task scheduling algorithm based on Tabu Search has been examined by Memari. P et al.[19]. The proposed algorithm is further enhanced by the implementation of the Fruit Fly Optimization Algorithm (FOA) and the Approximate Nearest Neighbor (ANN). The proposed and existing algorithms exhibit nearly identical time complexity in this work. Additionally, the dynamic demand generated by mobile devices is not considered when scheduling a group of tasks on a fog node. According to authors Cao, K. et al.[20], the location of the edge server is crucial for optimizing user latency, server energy consumption, and deployment costs. In order to optimize the expected response time of both the complete system and individual base stations, heterogeneous edge servers are implemented. In addition, the Integer Linear Programming (ILP) technique is employed to generate an optimal solution for the positioning of edge servers, and to accommodate user mobility, a game-theory-based scheme for base station remapping is implemented. There is a large overhead for remapping the solution at runtime in this study. Hästbacka, D. et al.[21] emphasize on data analysis. To integrate IIoT data and compose application services that are operated on both

the edge and the cloud, they developed a dynamic and interoperable architecture for edge-to-cloud service integration.The arrowhead framework is employed as a Software Oriented Architecture(SOA) to orchestrate and configure the composition of cloud and peripheral services at runtime. IoT platforms that are open-source have not yet attained the same degree of tool accessibility and user-friendliness as their proprietary counterparts. The end-to-end architecture developed by Akhound, N. et al.[22] integrates cloud, fog, and IoT layers with improved reaction times, resource efficiency, and availability. Additionally, they strive to extend the longevity, manageability, and data transfer rate of mobile IoT nodes. Here, the authors proposed a high-level hybrid architecture where fuzzy logic and a weighted linear combination-based clustering algorithm were used. In this work, a dynamic change in workload is observed due to mobile devices, which is not considered. The data's diversity, the size, and time-sensitivity present substantial challenges to the real-time collection, analysis, and decision-making processes. A framework for managing massive industrial data, facilitating online monitoring, and regulating smart manufacturing was put up by Saqlain, M. et al. in [23]. The five fundamental layers of the proposed structure are the database, application, middleware, network, and physical layers. An intelligent processing architecture based on edge computing that combines edge computing, fifth generation (5G), and Internet of Things (IoT) technologies was proposed by Bing, Z. et al. in [24]. In this work, individual edge node resources and task loads are not considered. It is advantageous to take advantage of the potential benefits of short-range, low-power communication technologies like Bluetooth, Zigbee, RFID, and Bluetooth Low Energy (BLE). Pattnaik, S. K. et al.[25] present BLE-based real-time location monitoring of employees using IoT. In this work, large signal transmission losses are observed, and hazardous situations are also not considered. Zhu, M. et al.[26] proposed a routing protocol to minimize unneeded packet forwarding for delay-sensitive applications in industrial IoT-based applications. In this work, the network topology remains a tree topology, and its components are transformed into a central star topology. In this protocol, additional memory is required for maintaining the neighbour node list, and additional overhead is added for updating the neighbour node list at runtime. Zhang, Y. et al.[27] proposed a framework, LsiA3CS, which mixes Deep Reinforcement Learning (DRL) and heuristic guidance and offers an interesting way to address the complicated task scheduling difficulties in large-scale Industrial Internet of Things (IIoT) environments. This study presents a technique in order to optimize workload and minimize communication delay, a sophisticated method of managing varied computational resources across distant edge clouds is demonstrated through the use of an Asynchronous Advantage Actor-Critic (A3C) algorithm and a Markov game-based model.Even though the suggested framework seems like a beneficial way to handle job scheduling issues in large-scale IIoT contexts, more investigation and testing are required to properly evaluate its efficacy, scalability, and suitability for use in actual industrial settings. Table 1, summarized several critical issues were identified in the previous works concerning the scheduling and resource management of tasks on fog/edge nodes, particularly in the context of dynamic workloads generated by mobile devices. The current approach fails to account for the fluctuating demand from these devices, resulting in significant overhead when remapping solutions at runtime is required. Additionally, the work highlights challenges related to data diversity, size, and time sensitivity, which complicate real-time collection, analysis, and decision-making processes. Individual edge node resources and task loads are also not addressed, further exacerbating the problem. The work also observes significant signal transmission losses and fails to consider hazardous situations, which could potentially affect the safety and reliability of the system. Moreover, the proposed protocol introduces additional memory requirements for maintaining the neighbor node list and incurs extra overhead for updating this list at runtime. By taking these factors into account, to push the boundaries of IIoT task scheduling and make it easier for distributed edge clouds to adopt coordinated, effective processing. In an industrial environment, overseeing and integrating a sizable number of IoT devices might present formidable obstacles. To maximize device utilization and data processing, strong architectures and effective scheduling algorithms are necessary. In order to avoid overload and guarantee proper operation, resource allocation algorithms must be properly implemented within the limitations of Internet of Things devices. To preserve connectivity and data flow, node mobility within the network should also be taken into account. Improving human-machine interaction and enabling smooth control and monitoring of industrial processes depend on the development of intuitive interfaces.

3. Problem Definition and Solution Overview

The initial portion of this section outlines the scheduling problem that was examined, followed by an overview of the proposed solution.

3.1. System Model

All the industrial machines and assets required to monitor and control the complete production process, equipped with data collection, limited storage, and computation capability, known as smart industrial devices or IIoT devices. These industrial devices are either mobile or fixed at a specific location.Table 2 represents a system model where the user sends a service request through various industrial applications.This request comprises numerous jobs, each of which contains a number of tasks. Each task is followed by a substantial number of instructions. The request can be defined as a collection of r jobs, when a user submits a service request to the system. Then, the edge system is defined by the following: the set of tasks (T_j) of jobs (J_i) (where $1 \leq i \leq r$), is sent to the edge system for processing and execution. User service request job J_i is defined as:

$$J_i = \{J_1, J_2, \ldots, J_r\} \quad (1 \leq i \leq r)$$

This job is divided into a set of t tasks.A task T_j is defined as:

$$T_j = \{T_1, T_2, \ldots, T_t\} \quad (1 \leq j \leq t)$$

and each task contains a large set of instructions.

$$DC_n^{J_1 \ldots r} = J_{n_1}, J_{n_2} \ldots\ldots\ldots J_{n_r} \tag{1}$$

set of job assigned to a DC_n

$$EM_m^{T_{r_1} \ldots r_t} = T_{r_1}^m, T_{r_2}^m \ldots\ldots\ldots T_{r_t}^m \tag{2}$$

set of task(t), of job r assigned to EN_m

$$Ins_{r_t}^{1 \ldots\ldots\ldots 10} \tag{3}$$

instructions no.1 to 10, task t of job r

3.2. Problem Definition

This section describes a task scheduling problem for the proposed edge-cloud and IIoT-based architecture. When a manufacturing system device (DN) or user generates and sends the service request (jobs) to a DC_n within its proximity, then jobs are forwarded to EN_m for locally executing the set of tasks and then offloaded to the cloud for remote execution if required. It is assumed that the placement locations of all DN except a few that are mobile, DC, and EN are fixed. So, to maximize device utilization and data processing, strong architectures and effective scheduling algorithms are necessary. The proposed architecture requires a scheduling algorithm that can efficiently decide at which EN_m the task execution load is forwarded and also to evenly distribute the task execution load amongst all the EN, such that the maximum rate at which tasks can be processed by an EN_m cannot be exceeded by the task arrival rate. It aims to optimize the average response time, i.e., the time to forward the computation load from DC_n to EN_m, then the required computation is done and the result is returned back to the user.

Table 1
Limitations of current approaches in task scheduling and resource allocation.

Ref.No	Authors	Methodology	Limitation	Evaluation Metric
[19]	Memari et al. (2022)	A novel task scheduling algorithm based on Tabu Search, ANN and FOA is proposed	Dynamic load due mobile device is not considered, Resource allocation algorithm is executing in constraint IoT device.	Execution time (2.743 s), Latency (14.02*10-5), Allocated memory(3.210MB), Cost function(3.236)
[20]	Cao et al. (2020)	Integer Linear Programming(ILP) and Game Theory	Significant time overhead for remapping solution at runtime, Managing large and more complex workflow is not considered. Response time(47.37%),	Response time fairness(71.60%), Speedup(24.08)
[21]	Hästbacka et al. (2021)	Architecture for interoperability and dynamic edge-cloud service integration for industrial IoT and Production Monitoring Applications.	Not yet equipped with all the essential tools and user-friendly interfaces of a specific IoT platform.	Total effort reduction (20%)
[22]	Akhound et al. (2022)	Fuzzy Logic , weighted linear combination and Firefly based scheduling techniques.	CH died much sooner, mobile IoT device is not considered, Dynamic change in workload due to mobile device is not considered.	Energy consumption(reduced), Packet loss (2.95), Packet delivery (91.42)
[23]	Saqlain et al. (2019)	Industrial Data Management System (IDMS) Framework Based on the Internet of Things (IoT).	Hardware part is not considered in this, No clear reduction in data size mentioned.	8.6 GB, 207.4 GB, 6220 GB, and 74,650 GB of real-time data were collected, resulting in a 33.1%, 22.1%, and 11.0% improvement in product quality.
[24]	Bing et al. (2023)	Distributive edge cloud architecture in intelligent coal mining systems.	Dynamic change in workload due to mobile device is not considered, Individual edge node resource and task load is not considered.	Threshold change (0.5-0.85),Fire response ratio(Tf_{rr}) increases, Algorithm updating time ratio (T_{utr}) less, Average data transmission volume ratio (V_{dtr}) less
[25]	Pattnaik et al. (2022)	BLE becon-based real-time location monitoring	Large Signal Transmission Loss, Other hazardous situations are not considered	Small visibility time, Real-time locations, Rescue process can be improved
[26]	Zhu et al. (2020)	Strategy of building a neighbor node list	Additional memory for maintaining the neighbor node list, Additional overhead	Time delay(shortest), Routing overhead(better), Delivery rate (90%)

Table 2
System model illustrating task and resource management.

Start:	Industrial devices generate data and send a service request to the edge node
Step 1:	Any data forwarding or processing job consists of t task.
Step 2:	The Device Coordinator (DC) executes the proposed scheduling algorithm for EN selection.
Step 3:	Each piece of data is forwarded, and computation will be done in the selected EN.
Step 4:	When the processing of the task at EN is completed, it puts the result together as a service response.
Step 5:	If required, EN sends its results to the cloud server in the upper layer, and then further processing is performed.
End:	The final result is returned to the user application, and the required action is produced.

3.3. Overview of Proposed Solution

Fig.1 represents the information flow in between the IIoT devices (edited version) [25]. These IIoT based manufacturing devices, machines and equipment's are BLE, Zigbee and RFID enabled to remain interconnected and communicate with each other to coordinate its activities, and multiple sensors are deployed with or between machines to enhance the visibility of processes, equipment's, and the status of products for better monitoring and control of the complete industrial system from anywhere. The data collected by these IoT devices facilitates improved decision-making and prompt notifications and alarm in the event of a fault. These data undergo initial processing at SBC which is RPi4 or node MCU based, including removing duplicates, arranging data in the required format, context-specific data segregation, and to display the required readings. All these data processing and visualization models require a numerous computing and storage resources. So due the limitations of IoT devices these data are forwarded to Local Server at the edge of the network for further advance processing by using CoAP protocol and then offloaded to the control office server or the cloud server for remote execution and storing the computation result if required.

4. Proposed Edge Based System Architecture and Response Time Model

The response time model is described in this section, which is preceded by an introduction to the proposed architecture.

4.1. System Architecture

Fig.2 illustrates the proposed system architecture. The proposed architecture consists of three different layers. The first layer, known as the device layer, consists of numerous mobile and static industrial assets, machines, and control panels as IIoT devices known as Device Nodes (DN). All these DN_s are equipped with data collection and communication techniques. In order to enhance device administration and communication in the challenging industrial environment, DN_s are organized in n clusters based on their proximity [28, 29]. In the device cluster, one is the dedicated cluster coordinator, known as the Device Coordinator (DC). The device coordinator receives data from all the DN in the cluster, aggregates it, stores it in a temporary data store, performs preliminary processing, and then forwards it to the next layer via the CoAP protocol [16].

Next is the edge computing layer, which contains m Edge Nodes ($EN_1 \ldots\ldots\ldots\ldots EN_m$), where (m is less than n) is the local server and one dedicated Resource Logger (RL) for storing the available resource details of all m Edge Node for allocating tasks to a suitable EN_m. And the last layer contains a cloud data center or database layer. The number of EN_s that a system host are heterogeneous in nature and is determined by the capacity of the RAM and hard disk, as well as the number of CPU cores and the speed of the CPU.

$$EN_m = \textbf{\{CPU core, CPU speed, RAM, Hardisc\}}$$

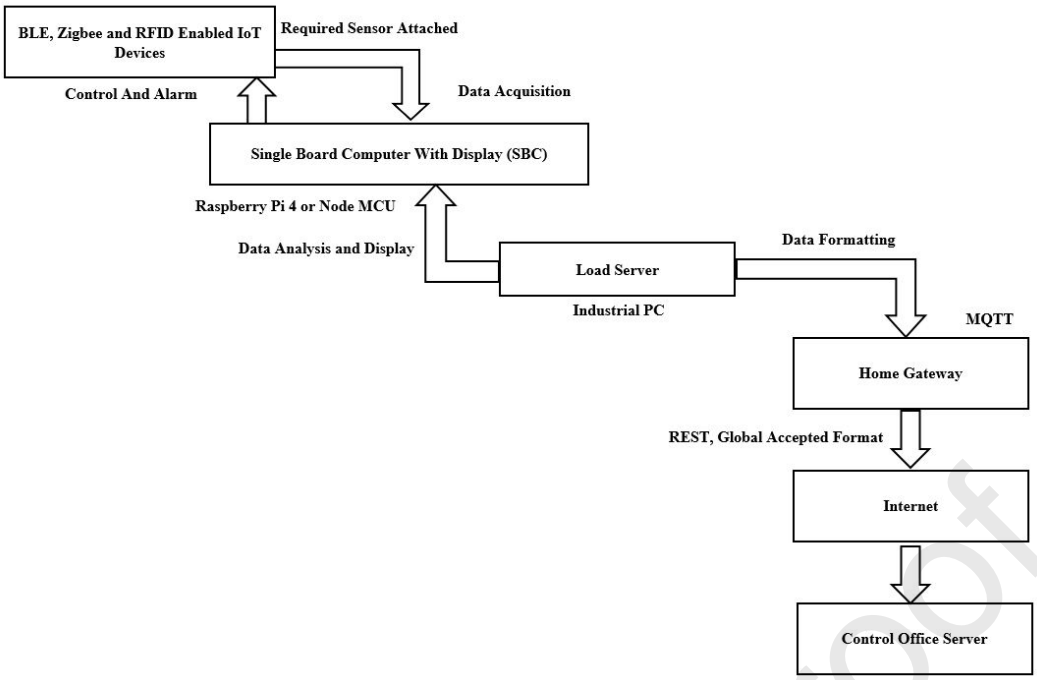


Fig. 1. Information Flow in between the different devices in an industrial setup.

All IIoT devices deployed on the factory floor for sensing, tracking the complete industrial environment, behavior monitoring, and facility management collect and communicate data to their respective Device Coordinators (DC_n). DC_n acts as a Programmable Logic Controller (PLC) for the set of IIoT devices within its proximity. Traditionally, IIoT depends on the uploading of collected or measured data to the cloud for analysis and storage. The process of moving data to the cloud, allocating the necessary processing resources, and subsequently returning the result to the factory floor typically involves communication delay, protocol conversion overhead, data transmission, and storage costs [30]. It is neither practicable nor desirable to transfer all data to the cloud in certain IIoT applications. According to the proposed architecture, DC_n initially forwards data to the appropriate EN_m. DC_n obtains resource details from RL and applies an RTT-based EN_m selection algorithm. EN_m transfers the computational load to the cloud for additional processing and storage if required.

4.2. Response Time Model

In the proposed architecture, the user sends a service request through various industrial applications. This request comprises numerous jobs, each of which contains a number of tasks. Each task is followed by a substantial number of instructions. The DC_n executes the task scheduling algorithm to assign the submitted tasks to the appropriate EN_m upon receiving the request. In order to ensure that the execution is completed within the designated time frame, or Time Bound (TB). Following the processing, the results that necessitate additional processing or require extended storage are transmitted to the cloud. Additionally, the computation results are combined and returned to the user in response to the requested service.

Following assumptions has been done to calculate the response time of the various nodes. Forward and return paths are assumed to have the same latency, Unless specifically analyzing congestion, queuing delays at intermediate routers are often omitted. , The overheads due to context switching or hardware interrupts are often considered negligible.,It is assumed that packet arrivals follow a Poisson distribution model.

4.2.1. Response Time of DN

The response time for a DN is the total time it takes to communicate the collected data to the DC_n for initial processing, then to the EN_m for further computation, and if necessary, to the cloud server, till receiving a response. Therefore, the response time for a DN is calculated as follows:

1. The time to upload the data to the communication link by DN is calculated as:
$$Trans_{delay}(DN) = \frac{L_{DN} * \sigma_t}{BN_{DN}} \tag{4}$$
where $Trans_{delay}$ is the transmission delay (or) time required to upload the data on the channel. L_{DN} is the size of the data to be transmitted from DN to DC_n; σ_t is channel noise during transmission; and BW_{DN} is channel strength between DN and DC_n.
2. The time to transmit the data from DN to DC_n is calculated as follows:
$$Prop_{delay}(DN) = \frac{2 * Dist_{DN-DC} * \sigma_p^2}{V_{electro}} + q_{delay} \tag{5}$$
$$q_{delay}(DC) = \frac{\lambda_{DC}}{\mu_{DC}} \tag{6}$$
therefore from (5) and (6), we get
$$\begin{aligned} Prop_{delay}(DN) &= \frac{2 * Dist_{DN-DC} * \sigma_p^2}{V_{electro}} + \frac{\lambda_{DC}}{\mu_{DC}} \\ &= \frac{2 * Dist_{DN-DC} * \sigma_p^2 * \mu_{DC} + \lambda_{DC} * V_{electro}}{V_{electro} * \mu_{DC}} \end{aligned} \tag{7}$$
Where $Dist_{DN-DC}$ is the distance between DN and DC, $V_{electro}$ is the signal strength, σ_p^2 is the channel noise during signal propagation, q_{delay} is the queuing delay, λ_{DC} is the task arrival rate at DC_n, and μ_{DC} is the task consumption rate of DC_n.
3. Therefore, there is a total communication delay to deliver the data from DN to DC_n
$$\begin{aligned} Delay_{commu} &= Trans_{Delay}(DN) + Propg_{Delay}(DN) \\ &= \frac{L_{DN} * \sigma_t}{BW_{DN}} + \frac{2 * Dist_{DN-DC} * \sigma_p^2 * \mu_{DC} + \lambda_{DC} * Vlectro}{Vlectro * \mu_{DC}} \end{aligned} \tag{8}$$

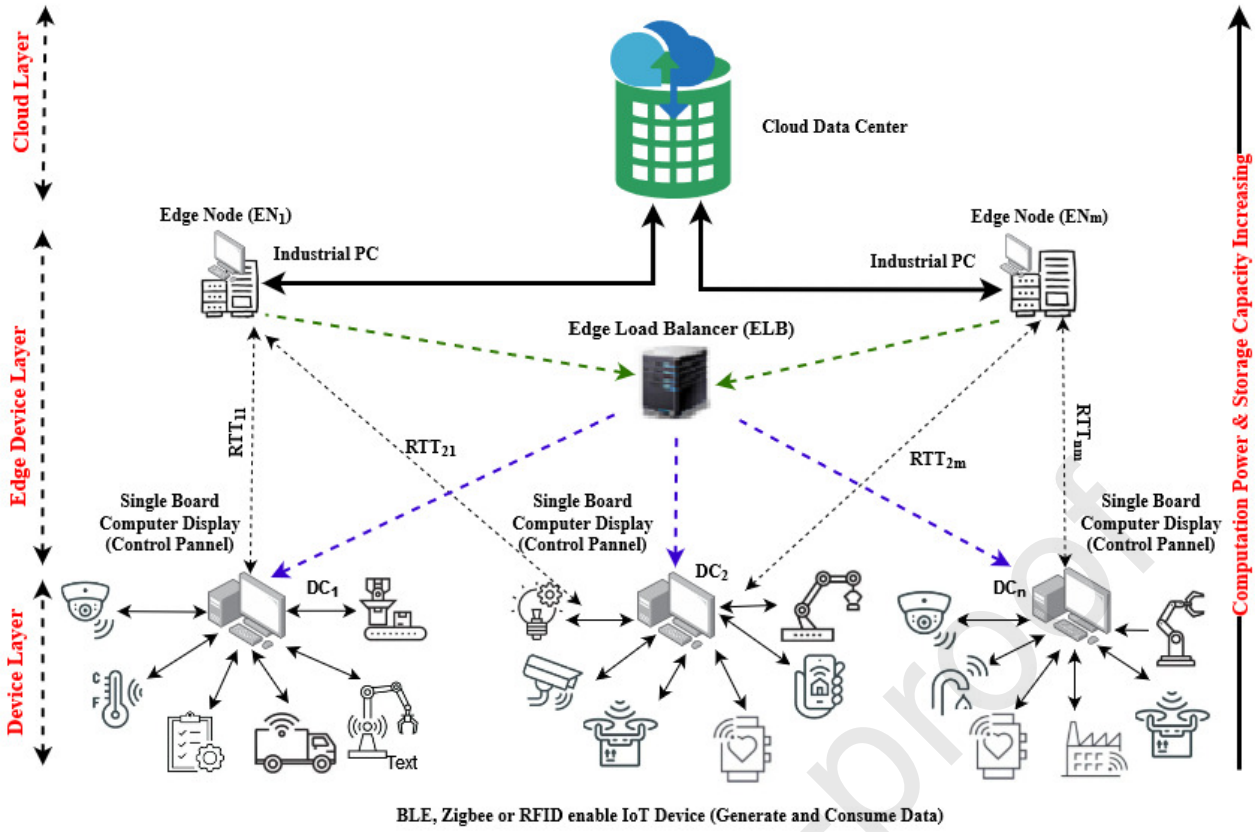


Fig. 2. Industrial Edge-Cloud Computing System Architecture.

4. The total computation delay at DC_n for initial data processing is given as:

$$Delay_{commu}(DC) = \frac{\Sigma_{i=1}^{r}\Sigma_{j=1}^{t}\Sigma_{l=1}^{L} Ins_{ij}^{i..l}(w_m * I_m + w_i * I_i + w_c * I_c * \eta_{cpu}(DC))}{N_{core(DC)}} \quad (9)$$

Where,

$\omega_m, \omega_i, \omega_c$ = weights for memory read instruction,input/output, and compute instructions, respectively.

$\text{Delay}_{\text{compu}}(DC_n)$ = Total computation delay at DC_n

I_m, I_i, I_c = Number of memory reads, input/output, compute instructions

$\eta_{\text{cpu}}(DC)$ = Processor frequency at DC

$\mathbb{N}_{\text{core}}(DC)$ = Number of processing units or cores in DC_n

$\text{Delay}_{\text{total}}(DN) = \text{Delay}_{\text{commu}}(DN) + \text{Delay}_{\text{compu}}(DC_n)$

5. Communication delay to deliver the data from DCn to ENm

$$Delay_{commu}(DC) = \text{(Time to Search for Neighbor ENs)} + \text{(Minimum RTT amongst all the ACKs received)} = K.dN + Min_{[RTT_{List}]} \quad (10)$$

6. Total Computation delay at ENm for data processing is given as:

$$Delay_{compu}(EN_m) = \frac{\Sigma_{i=1}^{r}\Sigma_{j=1}^{t}\Sigma_{l=1}^{L} Ins_{ij}^{i..l}(w_m * I_m + w_i * I_i + w_c * I_c * \eta cpu(EN_m))}{N_{core(EN_m)}} \quad (11)$$

$$Delay_{total}(DC_n) = Delay_{commu}(DC_n) + Delay_{compu.}(EN_m)$$

7. Finally the response time for a DN is given as:

$$Responce_{Time}(DN) = Delay_{Total}(DN) + Delay_{total}(DC_n)$$

$$Total_{delay}(EN_m^{Tr1...rt}) \leq TB(EN_m^{Tr1...rt})\textbf{(Condition to be monitor)} \quad (12)$$

where, TB is Time Bound within which we have to compute for a T_t of J_r.

5. Solution Approach and Algorithms

This section first explains the various approaches and their combinations. Then the algorithm that has been mentioned and its attributes are subsequently described. It relies on the COAP Confirmable (CON) and Acknowledgment (ACK) Message, which employs the Locality Sensitive Hashing (LSH) algorithm to determine the initial solution.

5.1. Task Scheduling Model

The process of job or task scheduling involves the assignment of priorities to the available tasks and the subsequent allocation of potential resources to the submitted tasks based on their priorities in order to reduce the execution time and cost and to achieve improved performance [31].

COAP Confirmable (CON) & Acknowledgment (ACK) Message: The Internet Engineering Task Force (IETF) [32] specifies the Constraint Application Protocol (CoAP), an application layer standard protocol, for IoT communication. The IETF designed CoAP to mimic the User Datagram Protocol (UDP) in order to reduce its frame size and accommodate transmission in a network with constraints. However, CoAP is also designed to support connection-oriented operation by means of an Acknowledgment (ACK) message similar to the Transmission Control Protocol (TCP) [33]. As shown in fig.3, ACK messages are sent in response to CON messages to acknowledge their receipt. They

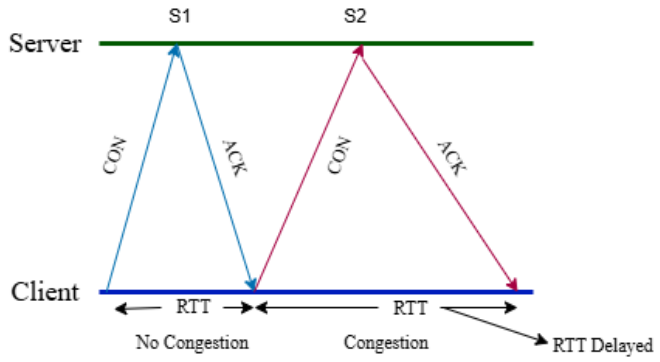


Fig. 3. CON and ACK Message communication between Client and Server.

indicate that the receipt has successfully received the message and is processing it. The Round-Trip Time (RTT) is the time period between a client node transmitting a CON message and receiving an ACK in response. During a normal situation (no congestion), the RTT period is relatively short or regular; otherwise, it takes longer to get back the ACK for a CON [15]. So the RTT can be used to analyse the situation of the server node, whether it is busy processing a large set of tasks or is free. This can help to decide where to forward the further computation task for execution, such that it can get processed with minimum delay within the required deadline, or TB When a significant number of IoT devices connect to an industrial network using modern communication standards, a problem arises. When IoT devices demand simultaneous and real-time transmissions to the server, the large number of connections causes congestion issues. This eventually causes adverse impacts on both throughput and transmission delay [34].

The Device Coordinator (DC_n) initially collects data, timestamps and segregates it, and compares it to its threshold value in the given industrial scenario. Then, the data is forwarded to the appropriate EN_m for further processing. So to manage congestion and uniformly distribute the task execution load on each individual EN, this work proposes a task scheduling algorithm to forward the computation load from DC_n to EN_m with minimal delay. The algorithm provides a technique to address the workload fluctuation specifically due to the movement of mobile devices in a network. And it is particularly possible in a network whose transmissions are regulated by the CoAP [35].

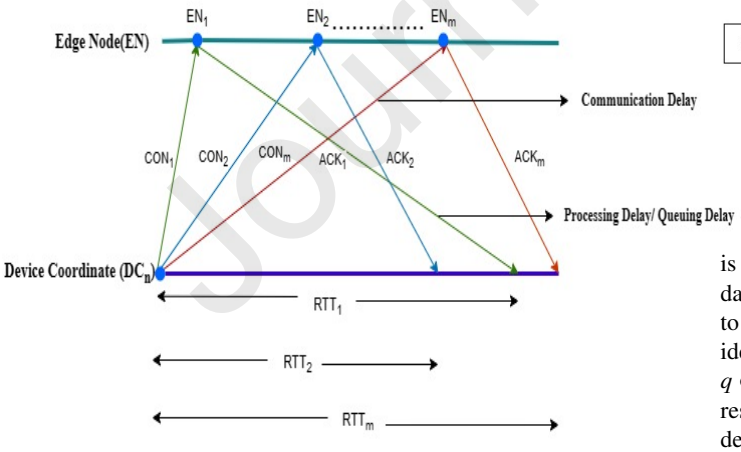


Fig. 4. CON and ACK Message communication between DC_n and EN.

As mentioned in the Fig.4 after collecting service requests from the associated DN_s, DC_n sends the CoAP-based CON messages to all the initial sets of EN_s, receives the ACK in return, and then calculates the respective RTT.

$$\text{RTT\_List} \longleftarrow [\text{ACK}_1(\text{RTT}), \text{ACK}_2(\text{RTT}), \ldots, \text{ACK}_m(\text{RTT})]$$

5.2. Initial Solution

In this work, instead of forwarding the CON message from the device coordinator (DC_n) to all the edge nodes (EN_m), for the RLRTT-based scheduling algorithm, a subset of all edge nodes is identified as the initial solution. Locality sensitive hashing is employed to achieve an optimized outcome and reduce the time required to identify an appropriate edge node, execution cost, and memory utilization. This technique is elaborated into the subsequent section.

Approximate nearest edge node selection solved by Locality Sensitive Hashing (LSH): Locality Sensitive Hashing (LSH) is a method by which similar items are hashed into the same bucket or set with high probability and dissimilar items are hashed together into the same set with low probability. This method is used for Nearest Neighbour Search (NNS), clustering, dimensional reduction, digital fingerprinting, and recommendation systems [36].

Motivation for using LSH: The two primary benefits of Locality Sensitive Hashing (LSH) are its theoretical guarantees on query accuracy and its sub-linear query performance (in terms of data capacity). Furthermore, LSH employs random hash functions that are data-independent, meaning that data properties such as data distribution are not required to generate these random hash values. Furthermore, the generation of these hash functions is not influenced by the distribution of the data. Therefore, these hash functions do not necessitate any modifications during runtime in applications where data is in motion or newer data is being introduced [37, 38]. State-of-the-art LSH techniques [39, 18, 40, 41] have resolved the issue of large index sizes that were a drawback of the original LSH index structure, which was necessary to achieve high query accuracy. Therefore, locality-sensitive hashing remains a critical technique for resolving the approximate nearest neighbour problem due to its small index sizes, rapid index maintenance, rapid query performance, and theoretical guarantees of query accuracy.

Overview of how LSH works for identifying similar node: As shown in Fig.5 nodes are represented as vectors in a high dimension space. Hash functions are applied to map nodes from the original space to a lower-dimensional hashed space [42]. Similar nodes are expected to be mapped to the same (or nearby) buckets with high probability. Nodes are grouped into buckets based on their hashed values. Similar nodes are likely to be found within the same (or nearby) bucket. Now only the sets of nodes that fall within the same buckets are compared based on the same threshold to find a final potential match that results in a similar set of nodes. The Nearest Neighbour Search (NNS) problem

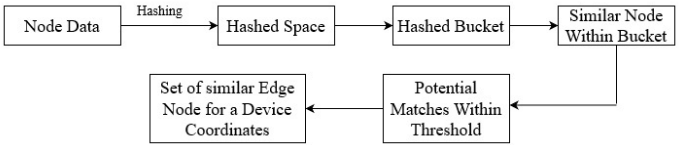


Fig. 5. Overview of how LSH works.

is defined as follows: We are interested in pre-processing an N-point dataset P in a d-dimensional Euclidean space R_d to promptly respond to k-nearest neighbour queries. In other words, we are interested in identifying the x data points from P that are the closest to a query point $q \in \mathbb{R}_d$. NNS is a fundamental geometric data structure problem that has resulted in numerous thrilling theoretical developments over the past decades, while also serving as a cornerstone of modern data analysis [43].

5.3. Proposed Neighbouring Node Selection Algorithm

In Algorithm 1, the initial set of nearest EN_s is identified for a particular DC_n (edited version of LSH) [43, 44]. As shown in fig.6 the space is divided into 2^k different regions by drawing k hyperplanes $(h_1, h_2, ...h_k)$.Each region contains points that do not exceed $\frac{N}{2^k}$.And all points inside the region R, are candidate solutions. For a typical query

point $q \in \mathbb{R}_d$, most of its nearest neighbours belong to the same region R. So compare q only to training points in the same region R; only those points are finally selected that are within the mentioned threshold. If there is a point y in R such that $d(x, y) \leq \text{th} \pm c$ then select y and add it to fcandidate_List. For a missed neighbour, repeat with different $(h_1, h_2, ...h_k)$ L times.

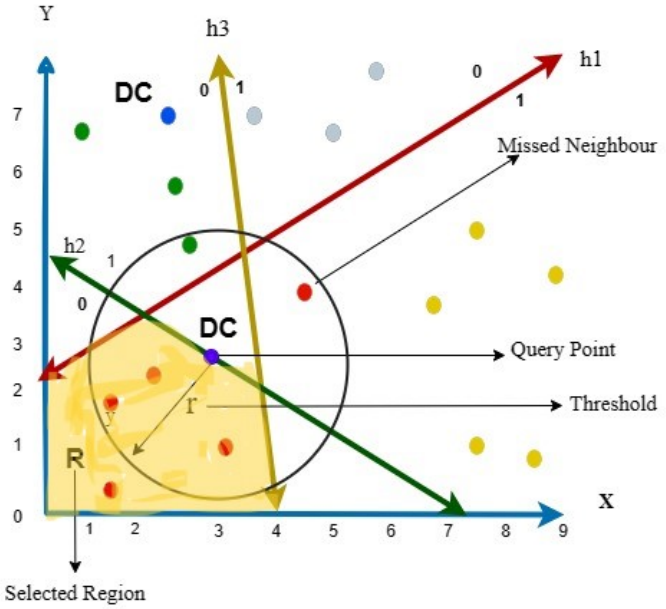


Fig. 6. Generation of hash code for data points.

$$\text{Complexity: } O(kd + \frac{dN}{2k}) \approx O(d \log N) < O(dN)$$

$$O(kd) \text{ to find region } R, \text{ where } k \ll N$$

$$\text{Compare to } \frac{N}{2k} \text{ points}$$

Where: N = Total no. nodes of all types on the place d = No. of dimension considered to represent the point/nodes k = No. of hyper line

5.4. Resource Logger to Evenly Distribute the Task Execution Load on ENs

A proposed model uses a dedicated *EN* as a Resource Logger (RL) to facilitate the equal distribution of task execution load among the EN_s at the edge of the industrial network for load balancing, which aims to reduce the processing delay at the nodes.Through the use of a continuous monitoring mechanism, the method collects real-time metrics from each Edge Node (EN) in the system, including CPU load, memory consumption, and network bandwidth.

Network traffic is distributed equally across a pool of resources that support an application through the process of load balancing. A load balancer, is a device that sits between the user and the server, handles a high volume of data traffic in a real-time and time-critical industrial environment, ensuring equal use of all server resources.

So, in this work, a DC_n is also used as a load balancer for a set of ENs in fcandidate_List identified as nearest neighbor nodes by the Algorithm 1. Dynamic Resource Based (DRB) method is used for this, in which RL analyzes the current loads on ENs. On RL, a specialized software application known as an agent is responsible for calculating the utilization of each EN resource, including its memory and computing capacity. And region (R_L^d) wise logs it in a table, where L is the number of regions $(1 \leq L \leq 2^k)$ and d is the number of dimensions considered for each data point. And each resource log table contains approximately $\frac{N}{2^k}$ entries and is shared with all DCs in the respective regions. Each

Algorithm 1 Neighbouring Node Selection Algorithm

Input: k random hyper plane $h_1, h_2, ...h_k$
Assumption: Consider manufacturing factory floor as plane and the different machinery and computer panels as node. There are three different category of node: IoT Node/Device Node(DN), Device Coordinator(DC) and Edge Node(EN). Let us consider total number of nodes on the plane is N, No. of DC = n, No. of EN = m, Therefore No. of DN = N - (n + m)
Output: Set of neighbour EN_s for DC_n

```
Draw k random hyper plane h_1, h_2, . . .. . . .h_k.
for each hyperplane (i=1 to k)
do
    Generate hash code for each point q as per the hyperplane h_i is drawn
    for each point or node on the plane(j=1 to N)
    do
        For each hyperplane one bit added in the hash code of every node
        Bit of has code = 0 if j.h_i < 0 and 1 if j.h_i > 0
        End
    End
Space sliced into 2^k regions
Each region(R) contain average N/2^k points
Select the points with similar hash code of query point q,(where q is DC_n)
scandidate_List ← points in region (R_L), where (1 ≤ L ≤ 2k)
Size Of The 1 ≤ scandidate_List ≤ N/2k
    for point y in scandidate_List
    do
        if (q, y) ≤ th ± c [ compared with required threshold ]
        do
        fcandidate_List ← y [ Add y in final candidate list ]
        End
    End
If missed neighbour go to step 1 and repeat until all neighbour node is identified
Return fcandidate list
End
```

individual EN_m resource availability detail (R_m) contains the following elements: CPU core, CPU speed, RAM, Hardisc therefore,

$$R_m = \beta_1 \eta_{core}(EN_m) \rho EN_m + \beta_2 Mem_r + \beta_3 Mem_{hd} \quad (13)$$

Where R_m is the resource availability detail of EN_m. $\beta_1, \beta_2, \beta_3$ are resource weight coefficients to measure the impact of a particular type of resource on computation. β_1 is 0, if the processor is busy, and 1 otherwise.β_2 and β_3 are 1, if $\frac{Mem_r}{Mem_{hd}} \leq 1GB$, otherwise 0, η_{core} is the number of processing cores, ρEN_m is the processing power of a processor, Mem_r is the RAM space available, and Mem_{hd} is the hard disk space available in an EN_m.The method can make well-informed scheduling decisions by using this data to identify patterns and variations in resource levels. For instance, the algorithm may give priority to rerouting jobs to less-utilized nodes in order to avoid possible bottlenecks if CPU or memory consumption on a specific EN hits a predetermined threshold. By choosing EN_s that are capable of handling data-intensive tasks, the algorithm can further optimize resource allocation in real time by minimizing latency through network bandwidth monitoring. Because of this ongoing monitoring strategy, the algorithm can be flexible, react to shifting circumstances, and guarantee excellent performance across the IIoT ecosystem.

5.5. Proposed RLRTT-ESA Algorithm

The proposed edge-cloud architecture aims to evenly distribute the task execution load among the EN_s and reduce the total process costs and energy use. To do this, the proposed RLRTT-ESA algorithm divides the submitted tasks at DC_n among the EN_s. This algorithm efficiently identifies EN_m where task execution load is minimum and quickly we can communicate and receive the response after processing. In this, the initial set of nearest EN_s is identified for a particular DC_n via the Algorithm 1. By sending the CoAP-based CON message to all the initial set of EN_s and receiving the ACK_s in return, calculate the respective RTT for every ACK received and add in RTT_List[x], where x is size of RTT_List whose value vary from $1 \leq x \leq \frac{N}{2^k}$. Also available resource

detail (R_m) is combined with the RTT (ACK_m) of EN_m in order to obtain the final score on the basis of which decision of best node could be made. The EN for which the final score is the minimum is selected as the best EN, and the task computation load is forwarded to that EN. The stopping condition in the proposed job scheduling algorithms is the maximum response time ($responsetime > deadline$), then the process is stopped and the task execution load is forwarded to the cloud platform. The proposed algorithm is described in Algorithm 2.

Algorithm 2 Proposed RLRTT-ESA Algorithm

Input: Initialize scandidate_List = $NULL$ fcandidate_List=NULL, BestEN=NULL, RTT_List = $NULL$, $ResdENs(RL)\_List$, $finalscore\_List$ = $NULL$

Assumption: Consider an array scandidate_List, fcandidate_List and RTT_List, $ResdEN_s(RL)\_List$ and $finalscore\_List$ of $size \leq \frac{N}{2k}$

For all DC_i from (i=1 to n)
do
  LSH is used to add nodes in scandidate_List(Call Algorithm 1)
  scandidate_List(DC_i)←points in region (RL), where ($1 \leq L \leq 2k$)
End For
For all DC_i from (i=1 to n)
do
  For all $EN_j(j = 1to\frac{N}{2^k})$ in scandidate_List(DC_i)
  do
    Calculate the Hamming distance(S_{ij}) of EN_j from DC_i
    if ($S_{ij} \leq th \pm c$) [where th ← Thresold distances to decide whether the EN_j is a neighbour node for DC_i and c=Adjustment factor]
      fcandidate_List(DC_i) ← scandidate_List(EN_j) [Add EN_j in , scandidate_List(DC_i) to fcandidate_List(DC_i)]
    else go to step 8
    End if
  End For
End For
fcandidate_List is forwarded to each respective DC with the resource availability detail list $ResdEN_s(R_L)$of each EN in a list by RL
$ResdEN_s(DC_n) = [R_1, R_2 \ldots\ldots\ldots\ldots R_m]$ [where, R_m is combined available resource detail of EN_m]
Send CON (CoAP) massage to all EN_s nodes in $fCandidate_{list}$
Receive ACK in response
Add Ack details with node id in a RTT_List
$RTT\_List \leftarrow [ACK_1(RTT), ACK_2(RTT), \ldots\ldots ACK_m(RTT)]$
For all DC_i from (i=1 to n)
do
  For all EN_j ($j = 1to\frac{N}{2k}$) in fcandidate_List(DC_i)
  do
    $f_j$score = $\frac{ACK_j(RTT)}{R_j}$
    finalscore_List(DC_i)← $f_j$score [Add the $f_j$score of EN_j in finalscore_List]
  End For
  BestEN(DC_i)←EN_m[Min [finalscore_List(DC_i)]]
  Return(BestEN(DC_i))
End For
Forword the task execution load to BestEN of all DC
END

6. Results Evolution and Insights

This section describes the implementation of the proposed scheduling algorithm, examines its outcomes, and compared them with conventional and previously proposed solutions.

6.1. Simulation Setup and Device Detail

The proposed architecture is simulated for a small IIoT-based industrial laboratory setup at a university to manage, control, and monitor various machinery and processes involved in the manufacturing system. The smart machines, robots, and equipment that have sensors for measuring their status and controlling the process are in the first layer of the infrastructure. These smart devices are connected to the industrial PLC known as the Device Coordinator (DC) within its proximity in the same layer. The DC sets its parameters to perform the initial processing of IIoT data, as detailed below. All the DC are connected to the Edge Node (EN) and RL in the second layer by using the proposed RLRTT-ESA scheduling algorithm. Lastly, the above layer establishes a connection between the EN from a variety of manufacturing configurations and the cloud, enabling it to undergo additional processing and data storage.

The following parameters were set during the simulation: There were 100–150 DN out of which 20–25 are DC, the arrival of the terminal tasks obeyed the Poisson distribution, there were 10–20 EN; the size of the tasks generated by the terminal ranged between 2 and 5 MB; the computing power of the DN and DC ranged between 5–10 MHz and 100–300 MHZ, respectively; and that of the EN ranged between 3–4 GHz. In this paper, simulation parameters are set based on several related works. [19, 20]. This work conducted extensive simulation experiments to validate the effectiveness of our proposed solution. The average task arrival rate and data volume of each base station are in the ranges of [4×106, 6×108] and [1, 100] MB, respectively [44, 45]. The communication capacity of each link is randomly selected in the range of [100, 1000] kB/s [46, 47]. The propagation rate of electromagnetic waves is set to 2×105 km/s [48, 20]. The parameters of the EN are listed in the above table.

6.2. Result Analysis

Real End-to-End service application tool within the Thingworx simulator implements the proposed IIoT system, allocating task in the EN according to the suggested RLRTT-ESA algorithm.To model device behavior and interactions, a number of settings can be set in the Thingworx simulator for IIoT systems. These include connectivity parameters like IP addresses and protocols to replicate network environments, and device attributes like temperature and pressure to depict real-time states. While data streams emulate the creation and transfer of data, event triggers can be set up to start operations based on predetermined criteria. Local processing and aggregation are modeled by edge device simulations, and application rules facilitate the automation of decision-making procedures such as predictive maintenance. Simulation time can be changed to reflect various time periods. These setups make it possible to run intricate simulations, which facilitate extensive testing and IIoT system optimization. In this simulation, the system is presented with 50–400 tasks, each containing $100 \times 10^3 - -400 \times 10^3$ instructions, for processing through various scenarios, as outlined below.

This work initially examined three different scenarios for determining whether the proposed idea of processing the maximum jobs at the edge of an industrial system is beneficial or not [20].

1. In Scenario 1 (SI), the IIoT device, or Device Node (DN), handles all processing.
2. Scenario 2 (SC) involves sole processing in the cloud.
3. According to the proposed model, Scenario 3 (SE) consists of collaborative processing at DC, maximum processing at EN, and cloud computing.

Number of Tasks/Jobs, number of instructions per task, associated data unit, execution time (ms), and communication and queuing delay (ms) are the metrics used to comprehensively compare the performance and behavior of the proposed system.Number of Tasks/Jobs represents the workload handled by the system. This metric helps assess the scalability of the system, showing how it performs under varying workloads. By varying the number of instructions, the system's ability to handle simple vs. complex tasks can be evaluated. Execution time reflecting computational efficiency provides insight into how well the system optimizes processing tasks of varying complexities and sizes. Helps evaluate real-time capabilities for systems with strict timing requirements. Communication delay accounts for the time taken to transmit data between nodes or devices, essential for distributed and IoT systems. It provides insights into the bandwidth usage and data processing efficiency.Queuing delay represents delays introduced by task scheduling and resource contention.

Table 3
Parameters of EN.

Parameters of EN	Values
Uplink BW:	1000 Kb/s.
Downlink BW:	300-1000 Kb/s
RAM	16 GB
System architecture	64 Bit
Operating system:	Windows Server
Energy consumption coefficient:	1 – 10
Operating system:	2 - 6
Computation capacity per node:	12 – 24 GHz

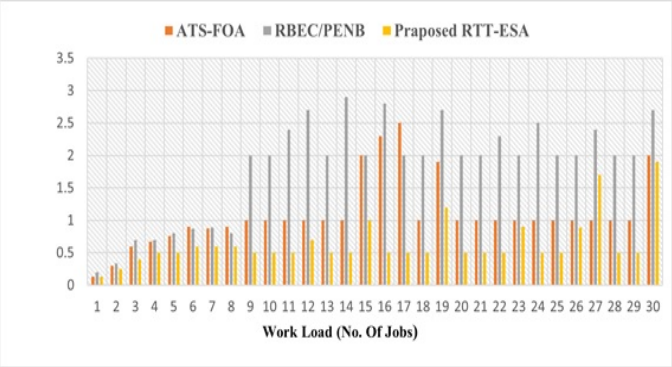


Fig. 7. Mean Response Time for a set of Jobs.

Table 4 presents the data for scenario 1 (SI). The inability to process Job2 and Job3 demonstrates a limitation in the existing resource configuration. It shows that there is a necessity for more powerful resources and resource management schemes to address a large number of complex tasks. The fact that all algorithms encounter substantial resource constraints when addressing larger tasks demonstrates the necessity for enhancements in resource allocation algorithms to address more tasks. However, Table 5 illustrates that Scenario 2 (SC) necessitates a significant communication and queuing delay for all algorithms to transfer data from an IIoT device to the cloud. This reduces the computation time but increases the overall response time. The computation and communication problems faced in scenarios 1 and 2 lead to the need for the proposed architecture mentioned in scenario 3. The proposed RLRTT-ESA consistently exhibits the smallest execution time as shown in Table 6, proving it the most efficient resource allocation and task scheduling algorithm. The least efficacy is indicated by the maximum execution times of RBEC/PENB, while ATS-FOA performs moderately well. Further, RLRTT-ESA exhibits the lowest communication and queuing delays for all task sizes, suggesting that processes are managed efficiently. ATS-FOA surpasses RBEC/PENB, but it does not function as efficiently as RLRTT-ESA. The most significant delays are observed in RBEC/PENB, which implies that communication and queuing are not managed efficiently. RLRTT-ESA is the most effective algorithm in all scenarios due to its minimal execution durations and communication/queuing delays. Table 7 depicts that the algorithm's performance, particularly the computation delay, is enhanced by RL, which notifies the DC of the available resources prior to transferring the computation tasks to it. The RLRTT-ESA algorithm is the optimal choice for managing tasks with varying instructions per task and associated data units, thereby guaranteeing the efficacy of both the execution and communication processes.

The efficiency of the proposed RLRTT-ESA algorithms is compared with the existing ATS-FOA and RBEC/PENB algorithms in Fig.7 across a variety of workloads, including 1 to 30 jobs. The mean response time is shown on the y-axis, while the number of jobs or workload is represented on the x-axis. The RLRTT-ESA algorithm repeatedly shows superior performance and the lowest response time across a variety of workloads. The performance of RBEC/PENB is subject to considerable variation across a variety of workloads, as illustrated

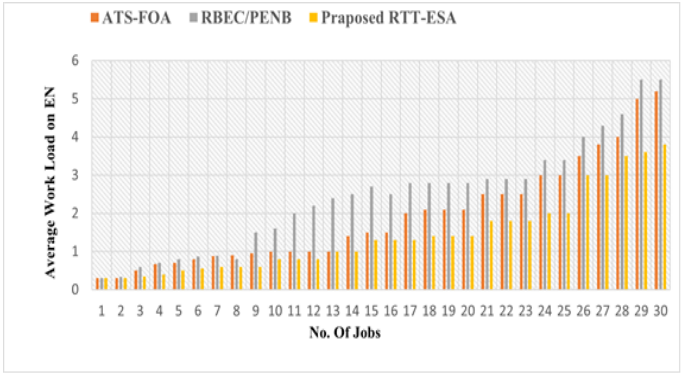


Fig. 8. Average Workload on ENs for a set of Jobs.

by several peaks in the figure that present high values. The proposed RLRTT-ESA and ATS-FOA show less variability and more stable performance across different workloads. The values for all algorithms tend to increase as the workload increases (moving from left to right on the x-axis), indicating a worsening of the performance metric with higher workloads. Despite the fact that RLRTT-ESA consistently maintains a lower value than the other algorithms, this shows its efficiency and robustness in managing larger number of jobs. The RLRTT-ESA algorithm demonstrates the maximum level of efficiency and stability across a variety of workloads in terms of the measured performance metrics. The summary suggests that the RLRTT-ESA algorithm is the most suitable option for scenarios that necessitate the efficient management of multiple jobs.

The average workload on EN for task numbers ranging from 1 to 30 is shown in Fig.8. The average workload on the EN is presented by the y-axis, while the number of tasks is represented by the x-axis. In comparison to the other two existing algorithms, the proposed RLRTT-ESA algorithm typically exhibits a lower average workload on the EN. This indicates that RLRTT-ESA manages resources more efficiently, which reduces load on the EN. The average workload on the EN for all algorithms increases as the number of tasks increases (moving from left to right on the x-axis). RLRTT-ESA consistently maintains a lower average workload than ATS-FOA and RBEC/PENB, suggesting improved scalability and resource management. The RLRTT-ESA algorithm that has been proposed is the most suitable option for scenarios that requires efficient and scalable resource management schemes, particularly as the number of tasks increases. It provides the most reliable and consistent performance, reducing the burden on the EN and resulting in an improved response time [18, 46].

The results show that the proposed RLRTT-ESA algorithm would result in substantial enhancements in system performance, particularly in environments with varying workloads and diverse jobs. Its capability to handle larger numbers of jobs with lower response times and delays, coupled with a better resource management strategy, makes it the optimal choice for enhancing the efficiency and scalability of the entire system.

7. Conclusion

The work introduced a new IIoT architecture based on edge-cloud computing that is specifically designed for manufacturing industries that require delay-sensitive monitoring and control systems. The research addressed the limitations of traditional cloud-based solutions and certain prior methods ATS-FOA and RBEC/PENB respectively, particularly in their capacity to efficiently manage large volumes of data with low latency. The proposed RLRTT-ESA algorithm shows better performance in terms of minimizing communication and computation delays. The total application response time is improved by 47.25% by the innovative task scheduling algorithm that is based on Resource Logger (RL) and Round-Trip Time (RTT). It guaranteed the effective management

Table 4
Result of Scenario 1 for ATS-FOA, RBEC/PENB and RLRTT-ESA Search Algorithm with 100×10^3 , 200×10^3 and 400×10^3 instruction/ task [19].

Algorithm /Jobs	No. of task/jobs	No. of instruction per task	Associate Data Unit (MB)	Execution Time in (msec)	Communication & Queuing Delay (msec)
ATS-FOA					
Job1	50	$100 * 10^3$	1.003	$20.94 * 10^2$	0.024
Job2	100	$200 * 10^3$	2.129	Unable to process task due to resource constraints	Unable to process task due to resource constraints
Job3	200	$400 * 10^3$	2.825	Unable to process task due to resource constraints	Unable to process task due to resource constraints
RBEC/PENB					
Job1	50	$100 * 10^3$	1.003	$22.14 * 10^2$	0.021
Job2	100	$200 * 10^3$	2.129	Unable to process task due to resource constraints	Unable to process task due to resource constraints
Job3	200	$400 * 10^3$	2.825	Unable to process task due to resource constraints	Unable to process task due to resource constraints
RLRTT-ESA (Proposed)					
Job1	50	$100 * 10^3$	1.003	$21.42 * 10^2$	0.025
Job2	100	$200 * 10^3$	2.129	Unable to process task due to resource constraints	Unable to process task due to resource constraints
Job3	200	$400 * 10^3$	2.825	Unable to process task due to resource constraints	Unable to process task due to resource constraints

Table 5
Result of Scenario 2 for ATS-FOA, RBEC/PENB and RLRTT-ESA Search Algorithm with 100×10^3 , 200×10^3 and 400×10^3 instruction/ task [19]

Algorithm /Jobs	No. of task/jobs	No. of instruction per task	Associate Data Unit (MB)	Execution Time in (msec)	Communication & Queuing Delay (msec)
ATS-FOA					
Job1	50	$100 * 10^3$	1.003	$2.72 * 10^2$	$36.098 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$4.12 * 10^2$	$39.232 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$6.23 * 10^2$	$45.113 * 10^-2$
RBEC/PENB					
Job1	50	$100 * 10^3$	1.003	$2.98 * 10^2$	$37.512 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$5.73 * 10^2$	$39.242 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$6.44 * 10^2$	$46.145 * 10^-2$
RLRTT-ESA (Proposed)					
Job1	50	$100 * 10^3$	1.003	$2.97 * 10^2$	$35.162 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$4.52 * 10^2$	$40.033 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$5.96 * 10^2$	$44.213 * 10^-2$

Table 6
Result of Scenario 3 for ATS-FOA, RBEC/PENB and RLRTT-ESA Search Algorithm with 100×10^3 , 200×10^3 and 400×10^3 instruction/ task Without RL[19]

Algorithm /Jobs	No. of task/jobs	No. of instruction per task	Associate Data Unit (MB)	Execution Time in (msec)	Communication & Queuing Delay (msec)
ATS-FOA					
Job1	50	$100 * 10^3$	1.003	$7.94 * 10^2$	$7.289 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$8.24 * 10^2$	$8.852 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$16.48 * 10^2$	$17.023 * 10^-2$
RBEC/PENB					
Job1	50	$100 * 10^3$	1.003	$8.98 * 10^2$	$9.282 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$9.58 * 10^2$	$9.304 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$18.56 * 10^2$	$18.586 * 10^-2$
RLRTT-ESA (Proposed)					
Job1	50	$100 * 10^3$	1.003	$7.42 * 10^2$	$5.325 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$7.97 * 10^2$	$5.827 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$14.99 * 10^2$	$11.323 * 10^-2$

Table 7
Result of Scenario 3 for ATS-FOA, RBEC/PENB and RLRTT-ESA Search Algorithm with 100×10^3 , 200×10^3 and 400×10^3 instruction/ task With RL[18]

Algorithm /Jobs	No. of task/jobs	No. of instruction per task	Associate Data Unit (MB)	Execution Time in (msec)	Communication & Queuing Delay (msec)
ATS-FOA					
Job1	50	$100 * 10^3$	1.003	$7.94 * 10^2$	$7.289 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$8.24 * 10^2$	$8.852 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$16.48 * 10^2$	$17.023 * 10^-2$
RBEC/PENB					
Job1	50	$100 * 10^3$	1.003	$8.98 * 10^2$	$9.282 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$9.58 * 10^2$	$9.304 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$18.56 * 10^2$	$18.586 * 10^-2$
RLRTT-ESA (Proposed)					
Job1	50	$100 * 10^3$	1.003	$6.32 * 10^2$	$5.325 * 10^-2$
Job2	100	$200 * 10^3$	2.129	$6.45 * 10^2$	$5.827 * 10^-2$
Job3	200	$400 * 10^3$	2.825	$11.23 * 10^2$	$11.323 * 10^-2$

of tasks of differing sizes by facilitating the efficient allocation of resources and the scheduling of jobs. The RLRTT-ESA algorithm shows enhanced scalability and resource management, ensured that the average workload on each Edge Nodes (EN) remains lower as the number of tasks increases. The results indicated that the RLRTT-ESA algorithm is the best choice for enhancing the efficiency, speed, and scalability of IIoT systems in manufacturing industries. The architecture effectively addressed the challenges of real-time processing, low latency, and efficient resource allocation and management, thereby ensured the sustainability and agility of industrial systems.The suggested approach has drawbacks in that there is no set number of hyperplanes that must be drawn in order to split the d-dimensional planes into distinct regions. The approach was dependent on particular environmental conditions and had limited scalability in large-scale installations.

The architecture may improve performance and efficiency by integrating with impending developments in edge computing, 5G networks, and artificial intelligence (AI). This proposed architecture will be more flexible in the future to be used in a variety of sectors, including smart cities, healthcare, and agriculture, which may stimulate more multidisciplinary research.

Declaration of interests: 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.

References

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, M. Ayyash, Internet of things: A survey on enabling technologies, protocols, and applications, IEEE communications surveys & tutorials 17 (4) (2015) 2347–2376.

[2] R. Minerva, A. Biru, D. Rotondi, Towards a definition of the internet of things (iot), IEEE Internet Initiative 1 (1) (2015) 1–86.

[3] R. A. R. A. Mouha, et al., Internet of things (iot), Journal of Data Analysis and Information Processing 9 (02) (2021) 77.

[4] H. Tyagi, R. Kumar, Cloud computing for iot, Internet of Things (IoT) Concepts and Applications (2020) 25–41.

[5] M. M. Sadeeq, N. M. Abdulkareem, S. R. Zeebaree, D. M. Ahmed, A. S. Sami, R. R. Zebari, Iot and cloud computing issues, challenges and opportunities: A review, Qubahan Academic Journal 1 (2) (2021) 1–7.

[6] S. Singh, Optimize cloud computations using edge computing, in: 2017 International Conference on Big Data, IoT and Data Science (BID), IEEE, 2017, pp. 49–53.

[7] W. Shi, G. Pallis, Z. Xu, Edge computing, Proceedings of the IEEE 107 (8) (2019) 1474–1481.

[8] M. Martínez Gil, Industry 4.0 in the theme park sector: Design of a real-time data acquisition system, Master's thesis, Universidad Carlos III de Madrid, https://e-archivo.uc3m.es/bitstreams/d31d7bbe-4783-4ef8-94a3-785cbd3d0bf6/download (2022).

[9] U. Sharma, M. Sharma, R. Singh, A. Chauhan, P. Pathak, Industrial internet of things: A survey, in: Emerging Trends in Data Driven Computing and Communications: Proceedings of DDCIoT 2021, Springer, 2021, pp. 291–298.

[10] R. Rosati, L. Romeo, G. Cecchini, F. Tonetto, P. Viti, A. Mancini, E. Frontoni, From knowledge-based to big data analytic model: a novel iot and machine learning based decision support system for predictive maintenance in industry 4.0, Journal of Intelligent Manufacturing 34 (1) (2023) 107–121.

[11] A. Pradeep, Integration of iot and industry 4.0: Revolutionizing industrial processes, in: International Conference on Advanced Communication and Intelligent Systems, Springer, 2023, pp. 85–96.

[12] H. Korala, D. Georgakopoulos, P. P. Jayaraman, A. Yavari, A survey of techniques for fulfilling the time-bound requirements of time-sensitive iot applications, ACM Computing Surveys (CSUR) 54 (11s) (2022) 1–36.

[13] S. Shukla, M. F. Hassan, D. C. Tran, R. Akbar, I. V. Paputungan, M. K. Khan, Improving latency in internet-of-things and cloud computing for real-time data transmission: a systematic literature review (slr), Cluster Computing (2023) 1–24.

[14] S. Shahzadi, M. Iqbal, T. Dagiuklas, Z. U. Qayyum, Multi-access edge computing: open issues, challenges and future perspectives, Journal of Cloud Computing 6 (2017) 1–13.

[15] P. Aimtongkham, P. Horkaew, C. So-In, An enhanced coap scheme using fuzzy logic with adaptive timeout for iot congestion control, IEEE Access 9 (2021) 58967–58981.

[16] S. Bolettieri, R. Bruno, Edge-centric resource allocation for heterogeneous iot applications using a coap-based broker, International Journal of Cloud Computing 9 (1) (2020) 28–54.

[17] E. Project, How edge computing and iot fit together, https://enterprisersproject.com/article/2021/3/how-edge-computing-and-iot-fit-together (2021).

[18] Q. Lv, W. Josephson, Z. Wang, M. Charikar, K. Li, A time-space efficient locality sensitive hashing method for similarity search in high dimensions, Tech. Rep. TR-759-06, Department of Computer Science, Princeton University, https://www.cs.princeton.edu/techreports/2006/759.pdf (2006).

[19] P. Memari, S. S. Mohammadi, F. Jolai, R. Tavakkoli-Moghaddam, A latency-aware task scheduling algorithm for allocating virtual machines in a cost-effective and time-sensitive fog-cloud architecture, The Journal of Supercomputing 78 (1) (2022) 93–122.

[20] K. Cao, L. Li, Y. Cui, T. Wei, S. Hu, Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing, IEEE Transactions on Industrial Informatics 17 (1) (2020) 494–503.

[21] D. Hästbacka, J. Halme, L. Barna, H. Hoikka, H. Pettinen, M. Larrañaga, M. Björkbom, H. Mesiä, A. Jaatinen, M. Elo, Dynamic edge and cloud service integration for industrial iot and production monitoring applications of industrial cyber-physical systems, IEEE Transactions on Industrial Informatics 18 (1) (2021) 498–508.

[22] N. Akhound, S. Adabi, A. Rezaee, A. M. Rahmani, Clustering of mobile iot nodes with support for scheduling of time-sensitive applications in fog and cloud layers, Cluster Computing 25 (5) (2022) 3531–3559.

[23] M. Saqlain, M. Piao, Y. Shim, J. Y. Lee, Framework of an iot-based industrial data management for smart manufacturing, Journal of Sensor and Actuator Networks 8 (2) (2019) 25.

[24] Z. Bing, X. Wang, Z. Dong, L. Dong, T. He, A novel edge computing architecture for intelligent coal mining system, Wireless Networks 29 (4) (2023) 1545–1554.

[25] S. K. Pattnaik, S. R. Samal, S. Bandopadhaya, K. Swain, S. Choudhury, J. K. Das, A. Mihovska, V. Poulkov, Future wireless communication technology towards 6g iot: An application-based analysis of iot in real-time location monitoring of employees inside underground mines by using ble, Sensors 22 (9) (2022) 3438.

[26] M. Zhu, L. Chang, N. Wang, I. You, A smart collaborative routing protocol for delay sensitive applications in industrial iot, IEEE Access 8 (2020) 20413–20427.

[27] Y. Zhang, H.-Y. Wei, Risk-aware cloud-edge computing framework for delay-sensitive industrial iots, IEEE Transactions on Network and Service Management 18 (3) (2021) 2659–2671.

[28] J. Coelho, L. Nogueira, Enabling processing power scalability with internet of things (iot) clusters, Electronics 11 (1) (2021) 81.

[29] L. Nogueira, J. Coelho, Self-organising clusters in edge computing, in: Intelligent Systems Applications in Software Engineering: Proceedings of 3rd Computational Methods in Systems and Software 2019, Vol. 1 3, Springer, 2019, pp. 320–332.

[30] H. Zhang, J. Shi, B. Deng, G. Jia, G. Han, L. Shu, Mcte: Minimizes task completion time and execution cost to optimize scheduling performance for smart grid cloud, IEEE Access 7 (2019) 134793–134803.

[31] Y. Guo, F. Luo, Z. Wang, H. Gan, M. Wu, H. Liu, Research on automated testing of in-vehicle time-sensitive network time synchronization mechanism, SAE International Journal of Connected and Automated Vehicles 7 (12-07-04-0027) (2024).

[32] K. H. Z. Shelby, C. Bormann, RFC 7252—The Constrained Application Protocol (CoAP), RFC Editor (2014).

[33] M. Gohar, J.-G. Choi, S.-J. Koh, Coap-based group mobility management protocol for the internet-of-things in wban environment, Future Generation Computer Systems 88 (2018) 309–318.

[34] J. Mišic, M. Z. Ali, V. B. Mišic, Architecture for iot domain with coap observe feature, IEEE Internet of Things Journal 5 (2) (2018) 1196–1205.

[35] R. Thakur, G. Sikka, U. Bansal, J. Giri, S. Mallik, Deadline-aware and energy efficient iot task scheduling using fuzzy logic in fog computing, Multimedia Tools and Applications (2024) 1–28.

[36] M. Datar, N. Immorlica, P. Indyk, V. S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the twentieth annual symposium on Computational geometry, 2004, pp. 253–262.

[37] O. Jafari, P. Maurya, P. Nagarkar, K. M. Islam, C. Crushev, A survey on locality sensitive hashing algorithms and their applications, arXiv preprint arXiv:2102.08942 (2021).

[38] F. Turrado García, L. J. García Villalba, A. L. Sandoval Orozco, F. D. Aranda Ruiz, A. Aguirre Juárez, T.-H. Kim, Locating similar names through locality sensitive hashing and graph theory, Multimedia Tools and Applications 78 (2019) 29853–29866.

[39] W. Liu, H. Wang, Y. Zhang, W. Wang, L. Qin, I-lsh: I/o efficient c-approximate nearest neighbor search in high-dimensional space, in: 2019 IEEE 35th International Conference on Data Engineering (ICDE), IEEE, 2019, pp. 1670–1673.

[40] D. Cai, A revisit of hashing algorithms for approximate nearest neighbor search, IEEE Transactions on Knowledge and Data Engineering 33 (6) (2019) 2337–2348.

[41] H. Farag, Č. Stefanović, Aoi-aware d2d communication in 5g-enabled smart grids: A multi-armed bandit approach, in: ICC 2024-IEEE International Conference on Communications, IEEE, 2024, pp. 3104–3109.

[42] A. Gharbi, Bi-directional adaptive enhanced a* algorithm for mobile robot navigation, Applied Computing and InformaticsHttps://www.emerald.com/insight/content/doi/10.1108/aci-12-2023-0195/full/html (2024).

[43] Q. Huang, J. Feng, Y. Zhang, Q. Fang, W. Ng, Query-aware locality-sensitive hashing for approximate nearest neighbor search, Proceedings of the VLDB Endowment 9 (1) (2015) 1–12.

[44] A. Gionis, P. Indyk, R. Motwani, et al., Similarity search in high dimensions via hashing, in: Vldb, Vol. 99, 1999, pp. 518–529.

[45] Y. Dong, P. Indyk, I. Razenshteyn, T. Wagner, Learning space partitions for nearest neighbor search, arXiv preprint arXiv:1901.08544 (2019).

[46] K. Cao, J. Zhou, P. Cong, L. Li, T. Wei, M. Chen, S. Hu, X. S. Hu, Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 38 (7) (2018) 1189–1202.

[47] K. Cao, G. Xu, J. Zhou, T. Wei, M. Chen, S. Hu, Qos-adaptive approximate real-time computation for mobility-aware iot lifetime optimization, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 38 (10) (2018) 1799–1810.

[48] J. Zhou, X. S. Hu, Y. Ma, J. Sun, T. Wei, S. Hu, Improving availability of multicore real-time systems suffering both permanent and transient faults, IEEE Transactions on Computers 68 (12) (2019) 1785–1801.