Meeting the Requirements of Internet of Things: The Promise of Edge Computing

Abhishek Hazra, *Member, IEEE*, Alakesh Kalita, *Member, IEEE*, and Mohan Gurusamy, *Senior Member, IEEE*

***Abstract*—Over the last few decades, Internet of Things (IoT) has become the spotlight area of research within the Industries and Academics. Primarily, IoT devices are characterized by small and nonscalable resources, including low processing capabilities, less internal memory, and short battery life. However, IoT applications demand extensive storage and faster response to ensure seamless and interoperable communication. Hence, Edge Computing and data/task offloading among the edge or cloud servers become promising, while also posing critical research challenges for edge-enabled large-scale IoT ecosystems. Several research activities have addressed the difficulties of determining an efficient and scalable data offloading strategy utilizing edge and cloud computing-supported technologies. This article focuses on the state-of-the-art edge IoT data offloading techniques, and optimization models in the heterogeneous IoT environment. We examine how edge and cloud-supported technologies can handle delay-sensitive IoT applications efficiently. Moreover, we introduce an IoT-based healthcare use case scenario to explain edge data execution and resource provisioning in IoT networks. Finally, we discuss several challenging issues and possible solutions to establish interoperable communication and computation for IoT applications.**

***Index Terms*—Cloud computing, computation offloading, connectivity, edge computing, Internet of Things (IoT).**

I. Introduction

THE Internet of Things (IoT) comprises an ever-increasing number of physical things connecting to the Internet at an unprecedented rate [1]. IoT is being widely used in intelligent transportation, innovative healthcare, industrial operations, smart agriculture, waste management, smart cities, and many more applications [2]. IoT technology can transform both living and nonliving entities available on the Earth's surface into smart object and connected them by advanced technologies, such as high-speed communication links, gateways, edge, and cloud [3]. Resource-constrained and smart IoT devices can run the applications locally which require minimal computation by exchanging information and coordinating decisions among themselves. Otherwise, the IoT devices need to offload their computational data to remote computing servers like edge servers or cloud data centers (CDCs) to achieve Quality-of-Service (QoS) requirements.

Vast amounts of data generated by IoT devices are sent to CDCs, such as Amazon Web Services (AWS), Google Cloud, ALTUS, Microsoft Azure, and IBM Cloud, for processing and data analytics. It is because cloud computing is the primary and widely used paradigm to meet the ever-increasing storage and processing-based service/application requirements. However, executing latency-sensitive IoT applications on a centralized cloud environment has the demerits due to the considerable physical distance between IoT devices and the cloud server [4]. For example, applications, including healthcare, connected vehicles, extended reality, and mining-related operations, require fast and on-device execution. CDCs are generally deployed globally and Intercloud communication and data migration increase the latency and response time when operation-dependent and delay-sensitive IoT applications are on the CDCs. The edge computing paradigm alleviates the disadvantages of the centralized cloud architecture, where local or adjacent edge devices (e.g., edge servers, small-cell towers, desktops, and base stations) can execute delay-sensitive applications locally to maximize the potential of the IoT ecosystem. Specifically, edge computing extends cloud computing by bringing storage, processing, communication, and networking closer to IoT devices. As a result, edge computing can organize and control local edge resources, further resulting in reduced latency, increased resource usage, and improved data privacy [5].

By shifting computing functionality from the cloud to edge networks, applications and services can enhance their response times. However, it is important to note that edge computing and cloud computing are distinct, despite their interdependence in terms of computation, storage, and applications [6]. Furthermore, in 2012, CISCO introduced the fog computing paradigm to bring computation, communication, and storage functionality closer to the network's edge [7]. However, in response to the demand for more data source-level execution, researchers are more interested in edge computing-enabled delay-sensitive IoT applications [5]. Efficient task offloading among edge devices is a highly challenging and extensively researched area. It necessitates the establishment of an optimal approach for offloading tasks, including the selection of which tasks to offload, determining the destination edge device

Manuscript received 8 November 2023; accepted 28 November 2023. Date of publication 5 December 2023; date of current version 21 February 2024. *(Corresponding author: Abhishek Hazra.)*

Abhishek Hazra is with the Communications and Networks Laboratory, Department of Electrical and Computer Engineering, National University of Singapore, Singapore, and also with the Department of Computer Science and Engineering, Indian Institute of Information Technology Sri City, Sri City 517646, India (e-mail: a.hazra@ieee.org).

Alakesh Kalita is with the ISTD Pillar, Singapore University of Technology and Design, Singapore (e-mail: alakesh_kalita@sutd.edu.sg).

Mohan Gurusamy is with the Communications and Networks Laboratory, Department of Electrical and Computer Engineering, National University of Singapore, Singapore (e-mail: gmohan@nus.edu.sg).

Digital Object Identifier 10.1109/JIOT.2023.3339492

2327-4662 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

for offloading, and identifying the appropriate timing for offloading. Numerous efforts have been dedicated to enhancing several objectives in the realm of task offloading, including energy efficiency, latency reduction, mobility enhancement, and cost optimization [6], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20]. Those works used several advanced techniques, such as game theory, queuing theory, and artificial intelligence (AI) to meet their desired objectives. However, there are many objectives and challenges that are still open and need to be investigated.

1) Why is edge-enabled data transmission indispensable for handling time-sensitive IoT applications?
2) Can we offload data near the edge networks for IoT applications without facing any challenges?
3) What are the state-of-the-art edge data offloading frameworks available for IoT applications?
4) Does remote data offloading solve the computation problem for IoT applications while considering several QoS objectives?

To fill the aforementioned research gaps, we offer a comprehensive literature review of the IoT–edge–cloud ecosystem concerning latency-critical IoT applications, where we explore edge-offloading techniques, optimization objectives, and offloading requirements for IoT applications. Furthermore, we introduce the importance of edge computing by examining an IoT-based smart healthcare scenario. In brief, the key contributions to this survey can be summarized as follows.

1) We conduct a comprehensive literature review of edge IoT execution architectures and compare their effectiveness in achieving various IoT objectives.
2) We discuss the need for making edge execution decisions and exploring various offloading architectures, including horizontal and vertical offloading techniques, especially for time-critical IoT applications.
3) We investigated edge-enabled computation offloading strategies and requirements for edge execution to facilitate device selection and decision making within the network. We also explored edge-enabled data offloading capabilities within a smart healthcare use case scenario.
4) We present the significance of state-of-the-art optimization algorithms in IoT data offloading within the context of edge execution and discuss their practical implementation.
5) Furthermore, we delved into core offloading challenges and presented potential solutions for adopting edge computation offloading technologies in IoT environments.

The remaining sections are organized as follows. In Section II, we introduced a brief summary of the existing literature. Section III discusses various computation offloading methods and strategies for real-time IoT applications. Section IV briefly analyzes computation offloading strategies, including offloading steps and suitable conditions. Section V introduces several remote execution optimization objectives. Several algorithmic aspects and their impact on optimizing IoT objectives are highlighted in Section VI. An IoT-based healthcare use-case scenario is also presented in Section VII. A summary of current research challenges and potential solutions is illustrated in Section VIII. Finally, we conclude our discussion by summarizing our viewpoint in Section IX. In Fig. 1, we summarize the content of this article.

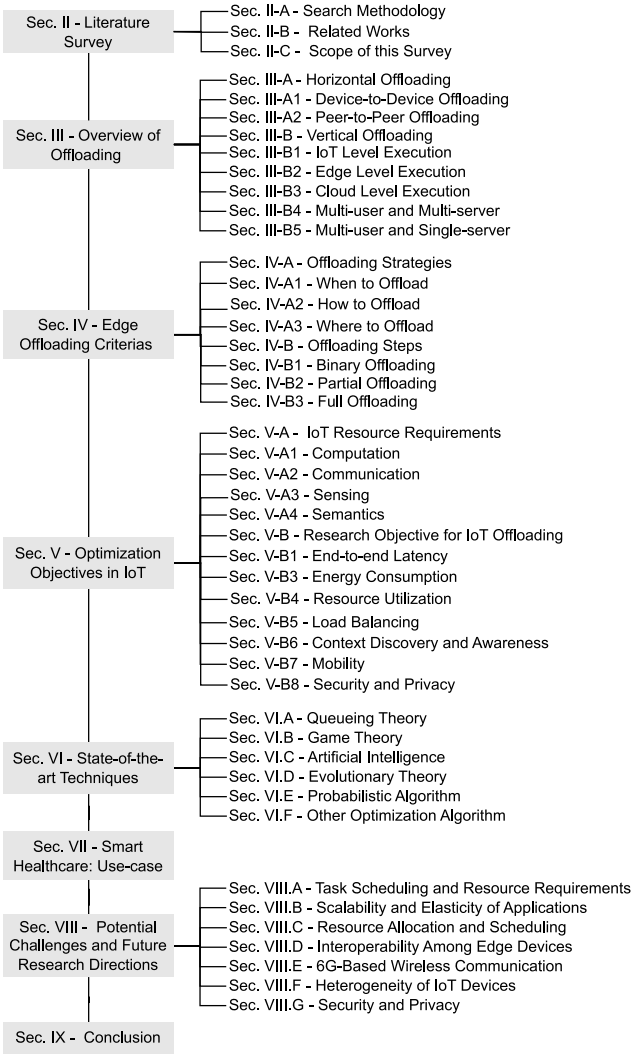

Fig. 1. Table of contents.

II. Literature Survey

This section offers a concise overview of the search process, the analysis of state-of-the-art survey papers, and the scope of this survey.

A. Search Methodology

Our study examined the literature concerning the necessity of edge computation for IoT applications by reviewing articles from four prominent scholarly databases (i.e., ACM Digital Library, Web of Knowledge, IEEE Xplore, and ScienceDirect) to gain insights into current research and future prospects. The keywords *edge computing*, *IoT*, and *offloading* yielded over 140 papers from 2015 to 2023. The selection of these keywords for this survey ensures a comprehensive and focused search across scholarly databases from 2015 to 2023 directly relevant to the research topic of edge computation offloading strategies. Among these published works, 59 articles were

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

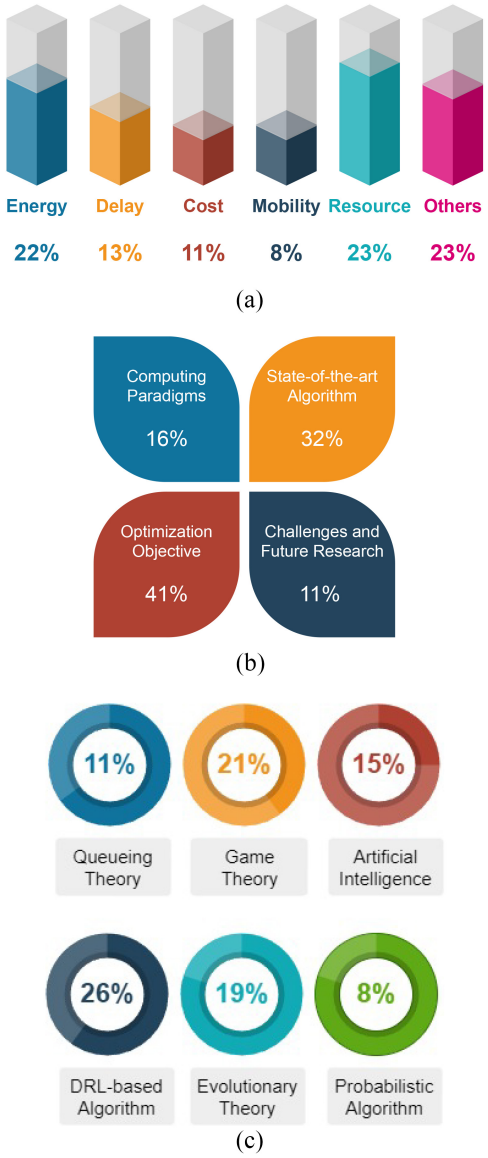

Fig. 2. Statistical analysis charts. (a) Research contributions related to optimization objectives. (b) Research contributions on various edge computing domains. (c) Research contributions related to optimization tools.

published between 2015 and 2018, with 18 (30%) of them addressing IoT architecture and the remaining 12 (20%) focusing on IoT offloading. Out of the 140 papers, a significant portion of 105 papers (60%) were published between 2019 and 2023, with a substantial number appearing in IEEE (84 papers) and Elsevier journals (34 papers). Through these research activities, we have the opportunity and responsibility to delineate our survey work from various perspectives. Specifically, this article discusses recent developments in collaborative IoT–edge–cloud evaluations and the associated challenges. We conduct an in-depth analysis and survey of the last four years of work (i.e., 2020, 2021, 2022, and 2023), which provides the foremost insights into IoT–edge–cloud architecture concerning edge computation offloading and execution issues. A concise statistical analysis of this survey is strategically depicted in Fig. 2, with Fig. 2(a) illustrating research contributions on various IoT objectives, Fig. 2(a) indicates the research directions (contributions and initiatives) on edge computing and Fig. 2(b) highlighting the optimization techniques widely employed to achieve IoT objectives.

B. Comparison With Existing Surveys

Over the past few years, several research articles have been published on IoT data offloading and remote data execution, where the majority of articles have addressed the requirement of remote computing/offloading in a simplistic manner [8], [9]. For example, Shakarami et al. [10] have presented various machine learning (ML) approaches for data processing in the mobile edge computing environment. The authors have also introduced several ML-driven state-of-the-art frameworks for computation optimization. A stochastic-based computation offloading strategy has been considered by Shakarami et al. [11], where the authors have discussed a number of performance metrics, delay dependencies, and offloading connections for edge networks. On the other hand, Yuan et al. [12] have presented a short overview for improving computation efficiency in vehicular edge networks. Aazam et al. [13] have furnished comprehensive literature on various IoT-enabled technologies and research opportunities for data processing in edge networks. Using the tactile Internet, Mukherjee et al. [14] have presented several challenges and limitations associated with intelligent workload distribution. Flores et al. [15] have inspected several mobile code optimization frameworks and concepts for supporting multitenancy for cloud-hosted applications. Similarly, other recent survey articles [4], [6], [16] have also focused on remote server execution techniques and challenges for collaborating IoT, edge, and cloud technology in one ecosystem. A summary of all the related survey papers is analyzed in Table I

Despite research in this domain spanning several decades, very few articles have visually represented edge offloading-related issues in the IoT ecosystem [17]. Notably, only a few research articles have considered the requirements of horizontal and vertical computing models for IoT or emphasized the significance of energy–delay tradeoffs and optimization techniques for edge IoT data execution. This article provides an analysis of the literature encompassing a wide range of objective and algorithmic aspects of edge IoT data execution strategies, accompanied by a use-case scenario illustrating how these aspects can be correlated to achieve an interoperable IoT ecosystem.

C. Scope of This Survey

This research endeavors to bridge existing gaps in understanding within the IoT–edge–cloud ecosystem, with a specific focus on the demands of latency-critical IoT applications. Our investigation mainly considers edge-offloading techniques, optimization goals, and offloading prerequisites for IoT applications. In addition, we explore the pivotal role of edge computing by scrutinizing a real-world case study in smart healthcare. Our contributions to this survey encompass several facets. First, we conduct an exhaustive literature review, critically evaluating the efficacy of diverse edge IoT execution

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

TABLE I
COMPARISON WITH EXISTING STANDARD SURVEYS AND TUTORIALS

Author and year	Survey objective	Survey scope for existing literature										
		Objective specific										Algorithm specified
		Energy	Data/ computation	Latency	Mobility	Priority	Cost	Processing time	Resource	Catching	QoS	
Azzedine *et al.*[19] 2019	Mobile cloud data offloading algorithms	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Jianyu *et al.*[20], 2019	Edge cloud data offloading algorithms	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Congfeng *et al.*[21], 2019	Edge oriented computation offloading	✓	✓	✗	✗	✗	✗	✗	✓	✗	✓	✗
Huan *et al.*[22], 2018	Vehicular data offloading in adhoc network	✗	✗	✓	✓	✗	✗	✗	✗	✗	✗	✗
Huan *et al.* [23] 2019	Data offloading in heter-ogeneous mobile network	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Huaming *et al.* [24], 2018	Decision based mobile cloud data offloading	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Sadri *et al.* [25], 2022	Data Reduction, IoT and edge computing network	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗
Kumari *et al.* [8], 2022	Offloading, objectives current and future direction	✓	✗	✓	✗	✗	✗	✗	✗	✗	✗	✗
Kong *et al.* [26], 2022	Edge computing and computation offloading	✗	✓	✗	✗	✗	✗	✗	✗	✗	✓	✓
Aazam *et al.* [27], 2021	Learning based mobile edge data offloading	✓	✗	✓	✗	✗	✗	✗	✗	✓	✗	✓
Aazam *et al.* [16], 2021	Short survey on edge computing, task offloading	✓	✗	✓	✓	✗	✓	✗	✓	✓	✓	✓
Flores *et al.* [15], 2021	Survey on various mobile code offloading techniques	✓	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗
Mukherjee *et al.* [14], 2021	Advancements of tactile internet for task offloading	✓	✗	✓	✓	✓	✗	✓	✗	✓	✗	✓
Aazam *et al.* [13], 2021	Review on IoT enabling technologies in edge computing	✓	✓	✓	✗	✗	✓	✗	✓	✓	✓	✗
Shakarami *et al.* [11], 2021	Computation offloading schemes for MEC environment	✗	✓	✓	✓	✗	✓	✓	✓	✗	✓	✓
Shakarami *et al.* [10], 2021	Survey on ML techniques for computation offloading	✗	✓	✓	✗	✓	✓	✗	✓	✓	✗	✗
Hazra *et al.* [27], 2023	Edge computing techniques for future IoT applications	✓	✓	✓	✗	✓	✗	✓	✓	✓	✗	✗
This Survey	**Promising Role of Edge Computing for IoT**	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

architectures in achieving various IoT objectives. Second, we explore the imperative of making informed decisions regarding edge execution and delve into a spectrum of offloading architectures, encompassing both horizontal and vertical offloading techniques, with particular emphasis on the exigencies of time-sensitive IoT applications. Third, we delve into the intricacies of edge-enabled computation offloading strategies and the requisites for streamlined device selection and decision making within the network, with a practical examination of edge-enabled data offloading in a smart healthcare context. Additionally, we emphasize the significance of contemporary optimization algorithms in the realm of IoT data offloading within edge execution environments, shedding light on their pragmatic implementation. Finally, we tackle core offloading challenges and propose prospective solutions, fostering the adoption of edge computation offloading technologies in IoT landscapes. Through this research, we aspire to provide comprehensive insights that will advance the deployment of IoT applications, particularly those with stringent latency demands, within the evolving IoT–edge–cloud convergence. The scope of this article is presented in Fig. 3.

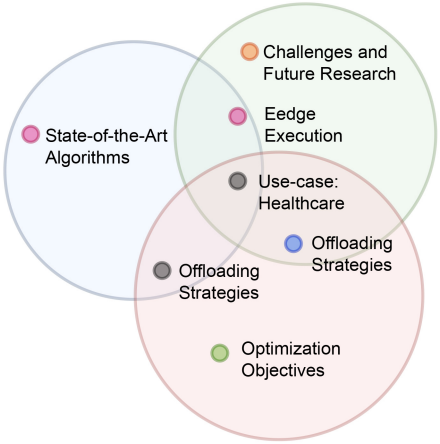

Fig. 3. Scope of this survey.

III. OVERVIEW OF TASK OFFLOADING IN EDGE-BASED IOT NETWORKS

The technique by which end devices transfer a portion or complete workload to a designated remote computing device for execution is called task offloading [27]. This method of transmitting data is known as *offloading*, and the percentage of offloaded tasks is referred to as the *offloading fraction*. End devices can optimize energy, cost, and delay by offloading tasks to remote computing devices. However, maintaining proper QoS metrics like throughput, performance, response time, and resource utilization is critical to the growth of IoT devices. Several decision-making factors are involved in selecting a suitable computing device for offloading, making it a complicated process. On the other hand, making the right task fraction and device selection may vary significantly based on data priority and application requirements [28]. After data processing at a selected computing device, the offloaded fraction is sent back to end devices using the original or alternative paths more efficiently.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

Furthermore, offloading objectives and offloading constraints are two attributes of edge task execution. Offloading objectives consider various QoS factors, including energy consumption, service latency, resource utilization, and more, while offloading constraints take into account bandwidth utilization, average network lifetime, and other metrics. These objectives are utilized in the design of a task execution technique to determine which criteria should be minimized. These metrics can also be assessed using edge task offloading techniques and cloud processing algorithms. To achieve the objectives of edge computation offloading, several requirements must be met, such as device size, data generation rate, service cost, latency, bandwidth usage, power consumption, and other application-specific needs. Based on the literature on remote computing architecture and data offloading strategies, we categorize edge data offloading into two classes: 1) horizontal offloading and 2) vertical offloading.

A. Horizontal Offloading

The primary requisite of most IoT applications includes real-time response and low delay tolerance. Hence, splitting the tasks into smaller indivisible subtasks as directed acyclic task graphs is desirable [29]. Parallel processing of such subtasks can decrease time and energy consumption compared to sequential processing methods. By parallelizing the task, it is possible to reduce response latency as tasks can be distributed across multiple processors [30]. Additionally, parallelization can reduce energy consumption as fewer resources are required for the same task. In this context, horizontal offloading of the task to nearby computing devices is one of the promising strategies to save energy, delay, and cost. The term *Horizontal offloading* refers to offloading these subtasks to service nodes in the same tier for further task processing [31]. This will eventually help minimize network burden and latency, a crucial factor for real-time IoT applications [32]. Moreover, this strategy can help to reduce energy consumption as the task can be offloaded to a nearby service node instead of processing it directly on the IoT device, reducing the overall cost of deploying IoT devices. Horizontal task offloading can be categorized into two types.

1) Device-to-Device Offloading: Device-to-device (D2D) offloading involves transferring computational tasks from one IoT device to another IoT device at the same level. In such scenarios, nearby devices can be employed if they possess idle resources or specialized capabilities suitable for processing specific tasks. D2D offloading proves particularly valuable when minimizing latency or conserving energy is paramount. For instance, when two devices are in proximity, they can share the task-processing load, resulting in energy savings. Moreover, D2D offloading can decrease latency by routing tasks to nearby devices with ample resources for faster processing.

2) Peer-to-Peer Offloading: Peer-to-Peer (P2P) offloading entails the distribution of workloads among devices within a network to optimize computational resources. In this context, peers refer to devices or nodes at similar hierarchical levels capable of collectively managing the computational workload. The adoption of P2P offloading can enhance overall system efficiency by harnessing the combined processing power and resources of multiple devices. Additionally, P2P offloading offers resilience against network disruptions, as tasks can be distributed among multiple peers. Furthermore, P2P offloading can reduce the reliance on dedicated infrastructure, such as servers, as peers can shoulder the load.

B. Vertical Offloading

Vertical offloading represents the most conventional offloading mechanism widely employed across various application domains. As the term implies, *vertical offloading* entails transferring the workload to available processing devices in different IoT–edge–cloud architectures. The need for nodes with sufficient computational capabilities becomes clear when local servers struggle to process tasks, even though they are horizontally distributed to sibling nodes [33]. Vertical offloading proves appealing as it offers the capability to handle computationally demanding tasks for edge nodes while simultaneously harnessing the supplementary resources of the cloud. This enables demanding applications to achieve higher accuracy and speed at the top levels of the architecture while taking advantage of the low latency and reliability of edge nodes. IoT devices typically use vertical task offloading to transfer tasks to edge nodes or cloud servers, providing superior resources and computational power. Such offloading aids IoT devices in diminishing latency and power consumption while enhancing privacy and security. Moreover, edge nodes and cloud servers tend to furnish more dependable services than IoT devices, owing to their heightened scalability and availability. Although this strategy may increase transmission delay for remote execution, it concurrently alleviates the computational burden on IoT-level and edge-level devices.

1) IoT-Level Execution: IoT devices, such as smartphones, smart gadgets, healthcare devices, or industrial sensing devices, are part of this layer that handles local data processing [34]. In telecommunications, the IoT device layer is also called the terminal layer. The key objective of this layer is to provide interaction between the physical and digital worlds using sensors and actuators. Initially, IoT devices sense and gather environmental information through sensors. Then based on the level of priority, this layer executes tasks on-device. Otherwise, it offloads tasks to remote computing devices through gateway devices [3]. Depending on the mode of communication, it can be done through wireless channels, 3G, 4G, WiFi Bluetooth, RFID, or near-field communication (NFC). This data offloading is due to IoT devices' limited storage and processing capabilities. One key advantage of IoT-level execution is the reduction of transmission delay to remote computing devices. On the other hand, instant decision making can be achieved through this layer. However, this layer may increase queue waiting time among computing devices. A flowchart of IoT-level execution is shown in Fig. 4.

2) Edge-Level Execution: The edge device layer is mainly responsible for collecting data from gateway devices and processing them in edge devices based on the priority of the

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

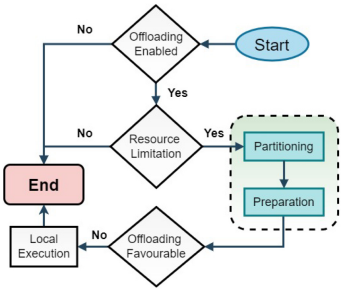

Fig. 4. Flowchart of IoT-level execution.

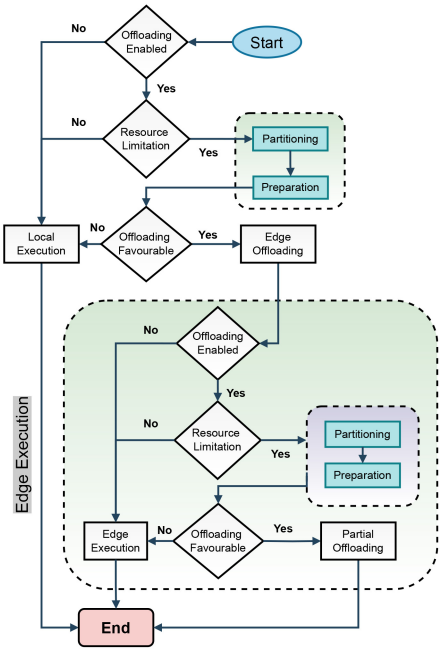

Fig. 5. Flowchart of edge-level computation.

data. This layer is an addition to the standard IoT-cloud architecture. The edge device layer mainly contains laptops, routers, switches, small-sized servers, and other intelligent devices for local computation and analysis [2]. The motivation for incorporating this layer is to introduce middle-level processing while optimizing energy, delay, and network bandwidth. For instance, significant accidents or machine failures can be prevented through IoT applications when appropriate actions are taken at the right time [35]. Among others, industrial applications are particularly sensitive, and corresponding industrial signals encompass high voltage, pulse rates, blood pressure, electric shock risks, and signals related to highly flammable gases. Moreover, latency can be minimized by analyzing the data close to the source. edge devices first filter the data based on data attributes [3]. Generally, high-priority tasks must be processed locally, low-priority tasks are offloaded to the cloud server, and middle-priority tasks are offloaded to nearby edge devices based on availability. Then, data is processed and transferred back to connected devices. If edge devices suffer storage and computation issues, they directly forward data to the cloud server. Fig. 5 shows the flowchart of edge-level execution for IoT-generated data.

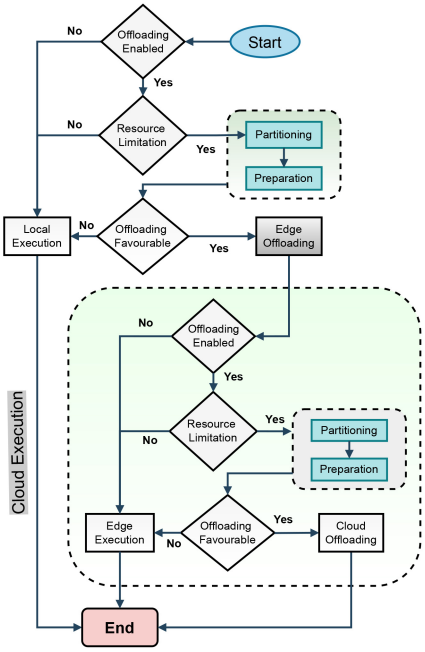

Fig. 6. Flowchart of cloud-level computation.

3) Cloud-Level Execution: The cloud layer represents a server layer executed over the Internet. Cloud services offer the capacity to store, manage, and process data using a remote server hosted on the Internet instead of a computer hard drive. This layer provides on-demand availability of computing resources for the long term [36]. Several top companies worldwide offer services to their users, such as Google Cloud, IBM Cloud, Amazon AWS, Microsoft Azure, Alibaba Cloud, etc. There are mainly four types of cloud services: public, private, hybrid, and community. The most prominent benefit of the cloud layer is the ability to present the pay-as-you-go model to customers. A main characteristic of this layer is access to a wide range of networks, resource pooling, measurement services, high security, rapid elasticity and scalability, and ease of maintenance [37]. Due to all these features, the cloud is a widely used technology for supporting several IoT applications. In the context of offloading, IoT applications typically transfer low-priority data to the cloud server, such as weather forecasting and agriculture data, for long-term processing and analysis, as shown in Fig. 6. However, in some cases, edge devices also transfer data to remote cloud servers due to the instant scalability functions of cloud computing. However, sending data to the cloud server incurs additional transmission time [4].

On the other hand, there could be multiple scenarios, such as multiuser–single server, single user–single server, or multiuser–multiserver. In this scenario, IoT devices offload tasks to one or multiple computing devices nearby for execution. For high availability and scalability, the IoT devices should communicate and collaborate. The IoT devices should also be able to store and process data locally. Additionally, the IoT devices should be able to coordinate with other devices in the cluster. In this server, we have discussed two popular task execution scenarios.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

4) Multiuser and Multiserver: While offloading data to remote computing devices, researchers consider several strategies while making any offloading decision. One of the simplest architectures for vertical task offloading is the multiuser and single-provider execution model [38]. In which multiple IoT devices send execution requests to multiple computing devices for execution. Similarly, computing devices can also receive multiple execution requests from IoT devices. However, the selection of remote computing devices is based on the offloading policy and other network/system-based parameters. If an offloading policy is binary, IoT devices can select one best suitable device among the available computing devices for execution [39]. This offloading policy can be based on several factors, such as energy consumption rate, data transmission cost, or speed at which IoT devices can process requests. The policy should also consider the security of IoT devices, as it is important to ensure that the data is not compromised or misused. On the other hand, partial offloading or partial co-offloading scenarios enable IoT devices to partition the data into several independent subtasks and distribute them to multiple remote computing devices [40]. This strategy allows IoT devices to collaborate with multiple computing devices hosted by different service providers. This model allows for the efficient use of resources, as tasks can be distributed across multiple IoT devices, and tasks can be easily balanced between different types of devices based on their capabilities and performance. Additionally, this model allows for efficient energy use, as tasks can be distributed across multiple devices, thus reducing the workload on any single device. However, this strategy sometimes creates execution delays due to the interdependence among the partitioned tasks.

5) Multiuser and Single Server: The multiuser single-server execution paradigm is widely utilized in distributed computing, especially in situations requiring extensive geographical reach and efficient utilization of computer resources. In such paradigm, multiple users or devices within a network establish connections with a single central computing server [41]. The purpose of this server is to function as a centralized hub for computational processes, enabling users to delegate their computational duties to it, as illustrated in Fig. 7. One notable benefit of this paradigm lies in its efficacy in encompassing vast regions without necessitating expensive infrastructure development. The accessibility of computing services to users is facilitated by the utilization of a central server, rendering it a highly appealing option for deploying applications, such as cellular networks or edge computing in geographically isolated areas.

However, it is important to acknowledge that this paradigm does have significant limitations. When multiple users are connected to a single server, a bottleneck issue arises due to inadequate computational capability to effectively manage incoming traffic. The issue of scalability is a significant problem in this particular environment. Moreover, this paradigm may face security concerns, including vulnerability to various forms of attacks, such as sybil attacks, jamming attacks, eavesdropping, and tampering. Despite these challenges, the multiuser single-server execution paradigm continues to be a desirable choice for situations where extensive coverage and efficient resource utilization are of utmost importance.

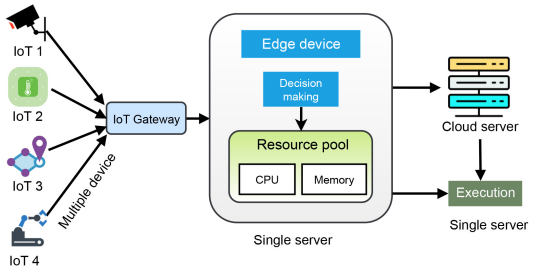

Fig. 7. Multiuser and single-server execution strategy.

IV. Edge Offloading Criteria

Edge data offloading refers to transferring latency-sensitive applications and data processing from IoT devices to a remote computing server through wireless transmission [42]. It is a method used in IoT to provide mobile assistance for IoT devices and to manage the increasing demands of big data applications in IoT. The process of edge task offloading involves considering various indicators to determine whether to transfer the task to the remote device. Several aspects of edge task offloading are discussed below.

A. Offloading Strategies

Offloading strategies encompass the methods and techniques used to redistribute or transfer computational data or load from a specific device. In the context of IoT, offloading involves wirelessly transmitting computational data to a remote computing server, enabling more efficient utilization of computing resources [43]. However, this domain faces several open research challenges concerning edge computing capabilities, offloading techniques, requirements, and, most importantly, when and where to offload IoT data. Offloading strategies are significant in the IoT domain as they improve the efficient utilization of computing resources, reduce latency, and enhance the overall user experience.

1) When to Offload: Offloading has gained significant attention due to its substantial benefits, including improved energy efficiency, enhanced performance, heightened reliability, and more effective utilization of contextual data. In conjunction with edge and cloud computing, facilitating data transfer from endpoint devices to a remote server offers an appealing strategy for addressing the aforementioned challenges. Consequently, selecting a remote computing device for data offloading becomes a viable option. Although the integration of edge/cloud computing technology into offloading decision making has undergone comprehensive scrutiny, it remains an ongoing area of research. It is evident that shifting the execution of data from end devices to remote computing servers can reduce execution delays and conserve energy usage. Nonetheless, remote execution is considered an optional choice rather than a mandatory requirement. This stems from the fact that the offloading process entails additional data transfer, which can potentially prolong transmission time and increase battery consumption related to

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

communication. Hence, decision delegation should be contemplated in scenarios involving substantial data transmission or limited bandwidth. This signifies that the offloading duration comprises the combined time spent on communication and computation on the remote computing server, and it is crucial for this duration to be shorter than the execution time on the IoT device to enhance performance. This means that when the remote server execution time, including uploading, processing, and downloading, is less than on-device execution, it is always advantageous to offload tasks. Consequently, migrating data to remote devices is advantageous rather than local execution, provided that the offloading condition is met.

2) How to Offload: Offloading tasks to a remote server (such as an edge device or cloud server) can be done in different scenarios, depending on the system requirements and the purpose of the task [44]. One common scenario for offloading tasks to a remote server is working with Docker, a platform for developing, shipping, and running container applications. This can be done by connecting to the remote server via SSH, adding a registered user, and giving it permission to run commands on the Docker host [17]. This can speed up the workflow and improve the efficiency of the development process. Debugging drivers and network devices can also benefit from transferring tasks to a remote server. Enabling or disabling task offload services with a registry key setting can help identify and resolve issues related to the functionality of drivers and network devices. Offloading tasks to a remote server can also be done for the purpose of increasing the efficiency of servers and improving application performance and security [45]. Offloading enables an application network infrastructure solution to perform tasks efficiently rather than consuming Web/server resources.

3) Where to Offload: IoT data can be offloaded to different locations depending on the specific use case and requirements. For example, offloading latency-sensitive applications (e.g., healthcare, industrial, and military operations) to edge devices is always preferable compared to cloud servers. On the other hand, compute-intensive applications (e.g., agriculture and weather forecasting) must be offloaded to the cloud server for long-term storage, processing, and analysis. However, edge servers are also restricted in terms of storage and processing capabilities. Therefore, in rush hour scenarios, edge servers have to prioritize tasks accordingly and send less sensitive data to the cloud server. Some common options include utilizing an opportunity-based network where users communicate directly without an infrastructure backbone through an opportunistic terminal-to-terminal (T2T) network [23]. This method can manage and transfer primary link data over T2T networks. On-premises deployment of Azure IoT Edge can facilitate consolidating operational data at scale and breaking up data silos. This method allows cloud-native workloads like AI or Azure services to be remotely deployed and managed. Data lakes can be used to move data from operational stores to a data lake to be analyzed efficiently using SQL queries. This process allows for building a low-cost and long-term archive of device data, which is useful for reporting and intelligence. A flowchart of the edge IoT data execution strategy is illustrated in Fig. 8.

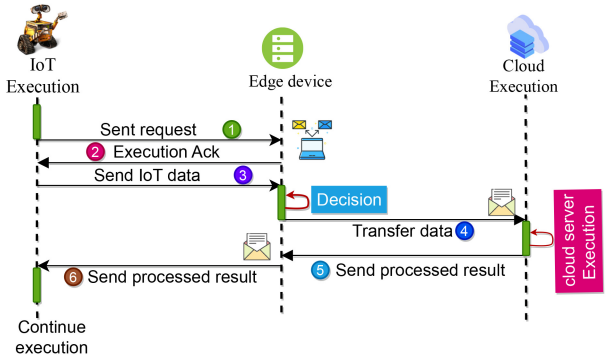

Fig. 8. Flowchart of edge data execution.

B. Offloading Steps

With the edge computing paradigm, real-time IoT applications can process data on resource-rich computing devices and manage data efficiently. There are three edge computing architectures for offloading data: 1) binary offloading; 2) partial offloading; and 3) complete offloading.

1) Binary Offloading: Binary offloading is a widely employed strategy in the IoT domain. It aims to establish an interoperable and efficient ecosystem by combining IoT devices, edge devices, and cloud servers [46]. The core intuition behind this model is the utilization of binary offloading decisions made by gateway devices for remote data execution, as illustrated in Fig. 9(a). Data from IoT devices are generally transferred to connected gateway devices, which act as the standard interface between IoT and computing devices due to their limited computational capabilities. In order to make offloading decisions, gateway devices implement heuristics, meta-heuristics, and ML techniques [23]. As a result of these binary offloading decisions, an appropriate computing device can be selected for the execution of a given task, such as one of the edge devices or cloud servers. An IoT-generated task can only be executed on a single computing device based on processing and storage capacity availability. In addition, this model facilitates the making of immediate decisions and supports the design of emergency decisions for delay-critical IoT applications.

2) Partial Offloading: Partial offloading is another data offloading model that aims to optimize the performance of IoT ecosystems. This model focuses on uniting edge devices and sharing excessive data among the connected devices in order to share data at the edge of the network and balance the load of the IoT network. When IoT devices request additional services from remote computing devices, the gateway device makes the offloading decision by selecting a computing device among the available options, such as edge or cloud servers, as illustrated in Fig. 9(b). Upon receiving a request, the cloud server immediately responds and begins execution, while edge devices check the availability of CPU and resource capacity before proceeding with task execution [47]. If an edge device requires additional resources, it obtains them from other computing devices (such as nearby edge servers or cloud servers). This strategy promotes interdevice collaboration and sharing of resources with underloaded edge devices. However, it should

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

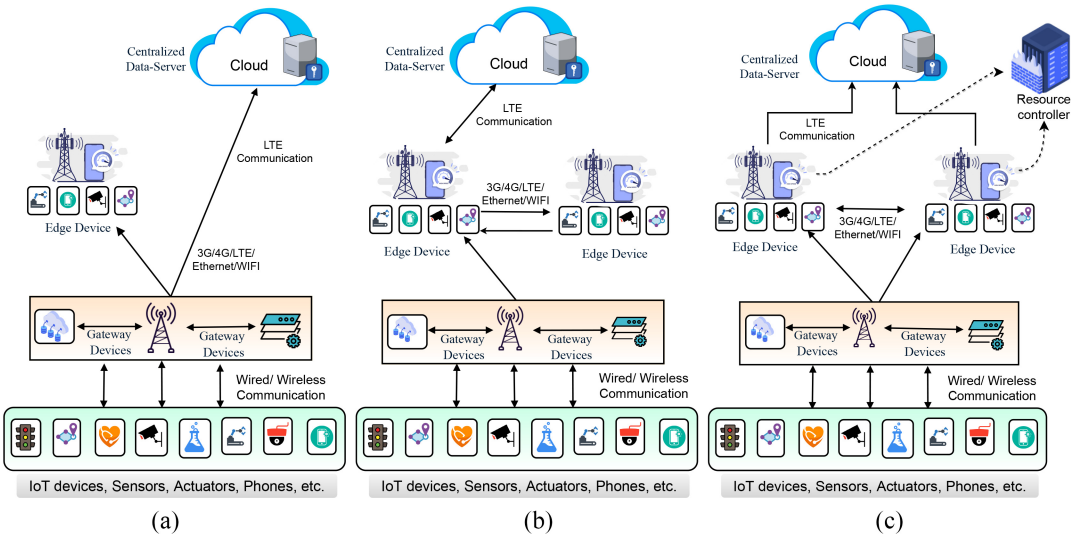

Fig. 9. Types of edge computing architecture. (a) Binary offloading architecture. (b) Partial offloading architecture. (c) Edge resource federation.

be noted that this approach also increases computation delay for task execution. One specific type of partial data offloading technique is partial co-offloading, where edge devices receive and send computation requests to nearby computing devices to improve cooperation capabilities between the devices. This way, edge devices from multiple service providers can share their excessive resources and increase resource utilization across the network.

3) Complete Offloading: In edge execution, computing and storage resources are brought close to the network edge to provide latency-critical services. However, service providers tend to establish their own private computing environments to cater to their users' individual needs, which limits the growth and dissemination of edge computing to meet a wide range of user needs. To address this limitation, an extension of the edge computing standard called edge resource federation has been proposed by many authors [48], [49], [50], which enables seamless collaboration and service provisioning between standalone edge-edge and cloud resource providers. Edge resource federation is a new approach to delivering latency-critical and computation-intensive services to traditional cloud users. The goal is to drive cloud services to users, such as central offices, base stations, or access points in order to reduce network congestion and latency. Service providers can now rent resources from infrastructure providers to host their services. Infrastructure providers typically deploy and manage several distributed computing systems at the network edge, each with computer and storage resources. They are responsible for service and resource supply.

However, edge computing paradigms have limited resource capacity and high maintenance costs. It is primarily due to the fact that infrastructure providers have the ability to build their own infrastructure in order to satisfy the needs of individual users. As each infrastructure provider manages and consumes its resources, standalone edge computing systems are frequently resource-constrained, particularly as the number of users increases. Furthermore, separate infrastructure providers may independently build edge nodes at the same location, resulting in uneven and underutilized edge resource utilization. This can impede the growth of the edge computing ecosystem due to ineffective resource management and service distribution. To address these issues, several studies propose edge resource federation as an integrated service delivery mechanism for the standard computing paradigm, which seamlessly integrates different infrastructure providers and cloud services to create a low-cost platform for end-users and edge resource federation.

In summary, cloud and edge computing offer advantages in their respective areas, but neither can simultaneously deliver high-quality services with minimal-cost resources. In contrast to cloud computing, edge computing and edge computing provide lower service latency and charge significant infrastructure and maintenance costs. On the other hand, cloud computing can provide highly scalable computation services at a lower cost but incurs high end-to-end execution latency. Therefore, edge computing and cloud computing can sustain real-time IoT applications.

Edge Resource Federation: The edge resource federation model, as opposed to the traditional edge/cloud architecture, comprises several essential components: IoT devices, edge servers, cloud servers, and a resource controller. IoT devices are primarily responsible for sensing and transmitting data over networks, while edge servers support latency-critical applications and provide additional resource support to user devices. In contrast, cloud servers offer long-term data storage, processing, and analysis. A key difference in this model is the inclusion of a controller device, as depicted in Fig. 9(c). The controller device collects all user service requests and facilitates resource assistance by gathering resources from different service providers. The objective is to gather and utilize all available edge resources to maximize profit for users and service providers. Additionally, overloaded edge servers can use the resource federation model to share the excessive workload with other nearby servers in the network environment.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

The edge resource federation, as described, is a trusted consortium authorized to control various service providers' resources. It aims to achieve scalability, low latency, and cost-efficiency by seamlessly integrating IoT–edge–cloud resources. From the perspective of IoT users, the key advantage of the edge resource federation model is its ability to efficiently handle dynamic service requests while optimizing service cost and delay. Furthermore, the main advantage of the edge resource federation model over traditional computing models is the unification of infrastructures from different service providers to improve service performance through optimized deployment strategies. This provides a significant benefit for service providers as well.

V. Optimization Objectives in IoT

Several research objectives and purposes are associated with edge computation offloading in IoT applications. These objectives must be taken into consideration with the utmost focus. This section starts by defining what edge computation offloading is and then discusses several research objectives.

A. IoT Resource Requirements

IoT devices require heterogeneous computing resources for processing and analyzing in-time environmental information. Here, we broadly classify the resources into hardware and software. Hardware resources, such as processors, communication media, sensors, actuators, microprocessors or microcontrollers, etc., assist in creating efficient and effective IoT devices for various real-time applications. However, software resources, such as real-time OS, communication protocol, data protocol, infrastructure protocol, semantic protocol, etc., benefit from maintaining, processing, analyzing, and managing real-time applications [51]. Based on their properties and working environment, IoT Resource Requirements can be divided into four categories: computation, communication, sensing, and semantics.

1) Computation: Nowadays, most IoT devices have some processing and internal storage capacity for executing small and delay-sensitive real-time applications locally [52]. Examples of IoT devices are Raspberry Pi, Intel Galileo, Friendly ARM, Arduino, Gadgeteer, UDOO, WiSense, etc. Each device has some processing components, such as a microcontroller, microprocessor, digital signal processing, FPGAs, and SOCs. Every IoT device relies upon its operating system (OS) to control and manage its heterogeneous components to operate effectively and efficiently. Various OS for resource-constrained IoT networks helpful for delay-sensitive IoT applications. Contiki-NG, RTOS, and OpenWSN are some of the most widely used OS for IoT devices, supporting a wide range of IoT applications. The Cooja simulator openly distributed along with the Contiki-NG OS allows developers and researchers to work and simulate various IoT and wireless sensor network applications [53]. Some other commonly used OS for IoT devices are RIoT OS [54], Lite OS [55], and TinyOS [56], which also consume less memory. Developers in the auto industry have formed the Open Auto Alliance with the collaboration of Google and are planning to develop a revamped Android platform with some enhancement features for improving the Internet of Vehicles paradigm for providing an effective response to users.[1]

IoT devices generate a wide variety of application data from delay-sensitive applications, which require many more resources for processing and storing the data than IoT devices. Platforms, such as cloud and edge, are extremely helpful in implementing IoT applications [31]. Edge devices are placed closer to IoT devices with limited resource capacities. However, cloud servers are primarily centralized and have sufficient resource capacity to execute any application. Devices that use IoT analyze the resource requirements and the QoS constraints of applications intelligently. They offload the workload to computational servers, such as cloud servers or edge nodes connected to a network.

2) Communication: IoT connects billions of heterogeneous objects and computation servers over the Internet and effectively and efficiently delivers various intelligent services. IoT devices generally forward request-based or resource-based applications to edge computational servers because they have shorter distances and consume less power while transmitting applications, such as edge devices. Bluetooth, WiFi, Z-wave, and IEEE 802.15.4 are all low-power communication links vital to the short-distance offloading strategy. They consume little power in noisy and lossy communications. WiFi technology uses radio waves to offload the data and information to the nearest computing servers (e.g., edge devices) within the range of 100 m for further computation [57]. WiFi technology allows various smart objects to communicate and exchange data or information without requiring a router. Bluetooth is another vital communication technology for exchanging data between smart objects over a small distance via a short-wavelength radio wave, which minimizes the end-to-end energy consumption of the communication link. Bluetooth 5.0 is the Bluetooth special interest group's latest technology for short-distance communication through IP connectivity. It is a reliable, high-speed network that uses very little power during data transmission.[2] A newly developed communication protocol is IEEE 802.15.4, which specifies the data link and physical layer for offloading data through a reliable and scalable network with the minimum power consumption [58].

NFC and ultrawide bandwidth (UWB) are a group of high-speed communication technologies that enable reliable and low-power short-distance communication. RFID technology was first used to communicate between machines by researchers and developers. An RFID reader generates and transmits a query signal to the tag of an object. It also receives a reflected signal from the object tag, passed to a database. Based on the reflected signal, the database is connected to computing devices that identify the object within a range of 10-200 m [59]. In general, the communication range of the NFC protocol between active and passive readers or two active readers is up to 10 cm [60]. This protocol uses a high-frequency band at 13.56 MHz and offloads the data at 424 Kb/s. Finally, the UWB protocol is developed for

[1]Open Auto Alliance, 2018, Available: http://www.openautoalliance.net/.
[2]http://www.bluetooth.com/Pages/Press-Releases-Detail.aspx

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

communicating between two closely placed objects with a low range of convergence [61]. This protocol uses low power and high bandwidth to exchange information between objects. IoT devices employ long-term evolution (LTE) technology to communicate over long distances and offload data and information to the cloud data center. LTE technology is most suitable for high-speed data transfer between UMTS/GSM network technology-based mobile devices [62]. This technology supports fast data transfer for broadcasting and multicasting. The advanced version of LTE technology, namely, LTE-A, supports higher bandwidth up to 100 MHz, lower latency, increased throughput, extended coverage, and uplink and downlink spatial multiplexing [63].

3) Sensing: Nowadays, IoT is an exciting technology that allows users to gather data from their surroundings in real time and analyze those data on their edge computing devices or cloud computing servers for decision making. Sensors in the IoT collect data from various objects connected to them and transmit it to a data warehouse or computational server. This is done to take appropriate actions based on multiple actuators or motors. Sensor data and information are analyzed using heuristic, meta-heuristic, and fuzzy-based strategies to determine the action based on the required services. The IoT sensor is a smart object that can measure physical phenomena, such as temperature, humidity, pressure, motion, etc., and helps detect environmental changes. On the other hand, the IoT actuator is a mechanism for tuning energy to signal. After analyzing the sensed data, the actuator allows specific actions to be performed by the computational server or data warehouse. Some companies like Revolv, Wemo, Smart Things, etc., offer various smart hubs and mobile applications that enable users to control and monitor hundreds to thousands of smart objects and applications that are surrounded by the environment using their smart phones [64], [65]. Single-board computers are embedded in various sensors and produce multiple types of IoT devices, such as Beagle Bone Black, Raspberry Pi, Arduino, etc., which support the TCP/IP protocol for data transmission through a reliable and secure network. Those devices mainly connect to a central management portal for analyzing and processing the data that users request.

4) Semantics: Semantic technology is an integral part of IoT technology, which can extract knowledge accurately and smartly based on predefined rules or historical data sets by employing heuristic or meta-heuristic algorithms to provide specific actions. The semantic approach allows data to be analyzed and recognized more accurately, which assists in making the right decision for accurate services based on actuators [80]. Using semantics, IoT technology can make the right decisions and send appropriate requests to the correct resources. The Web Ontology Language and the Resource Description Framework are the basic requirements for developing semantic technology for IoT devices. The World Wide Web consortium developed efficient XML interchange (EXI) for developing smart semantics for IoT devices in 2011 [81]. EXI is essential for IoT devices for smart semantics in a resource-constrained environment due to its smart design, which helps optimize XML applications. Moreover, EXI technology minimizes the required bandwidth for data and information exchange without affecting the related resources of IoT devices. This includes power consumption for processing, battery power, memory, and data code size. EXI technology converts XML messages into their corresponding binary form to minimize overall bandwidth.

B. Research Objective for IoT Offloading

IoT technology requires addressing several evolving research challenges to enable effective and efficient user applications. IoT devices generate an enormous amount of real-time data and applications that must be processed and analyzed effectively to meet various QoS objectives, such as minimum latency, computational time, energy consumption, maximum resource utilization, etc.

1) End-to-End Latency: The latency issue is significant in IoT applications, especially in delay-sensitive applications resulting from critical and complex environments, such as industrial automation, mining operations, transportation, and railway management [82]. These delay-sensitive applications must be executed locally or on computational servers near the IoT devices, such as edge servers, gateway devices, routers, mobile phones, tabs, laptops, etc. This strategy helps minimize the total communication overhead and end-to-end latency of IoT applications while transferring them to remote computing devices. Latency also highlights the importance of round-trip time and the computation time of IoT devices. Minimum latency minimizes the total round-trip time and computation time in the network. Precisely, computation delay pertains to the duration required for data processing and the time complexity of the running algorithm, while communication delay pertains to the duration needed for data transmission within a network. Round trip time, conversely, quantifies the duration for data to traverse from its origin to a designated endpoint and subsequently return. On the other hand, end-to-end latency comprehensively encapsulates the complete duration for data to traverse from its source to its intended destination, encompassing the cumulative effect of all encountered delays throughout this transmission process.

Edge computing technologies can reduce the latency of real-time IoT applications by processing those devices separately or distributing them among multiple devices in a distributed environment. However, a CDC may be assigned to applications that require maximum resource capacity. The centralized nature of the CDC increases transfer delay when data is sent to remote computing devices but still meets application resource needs. Simply put, when important data needs to be processed quickly, it is usually done on edge servers. However, if the data need to be transferred to another edge server during busy hour, it can cause a delay in processing. This delay can be dangerous in healthcare- and mining-based industries where data sensitivity is very important. To prevent this, the servers must process the data close to the edge while meeting the time requirements. Table III illustrates the current studies considering latency as the optimization objective.

The performance of IoT applications is highly dependent upon the duration of their transmission and processing times. In particular, the transmission time of such applications is

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

TABLE II
COMPARISON OF EXISTING WORKS WHERE OBJECTIVE IS LATENCY

Contribution	Environment	Types of offloading	Constraints	Distance measure	Proposed technique	Task partition	Execution level	Task priority	System model	Server type	Advantage	Research gap
Wu et al. 2018 [69]	Dynamic	Heterogeneous Networks	Connectivity, delay	Multi-Hop	ERWP	✗	Cloud	✗	Mobile cloud	Cloudlet-VM	Mobile data offloading in the heterogeneous network	Priority based intelligent decision making system
Gao et al. 2017 [70]	Wi-Fi networks	Wi-Fi	Deadline, capacity	Multi-hop	FDO, HDO, and NDO	✗	Wi-Fi APs	✓	Mobile cloud	Cloud Server	Deadline based mobile data offloading	Energy-delay tradeoff for dynaic networks
Hu et al. 2012 [71]	Target area	Wi-Fi	Sojourn time	Multi-Hop	Stochastic -Geometry	✗	APS	✗	IEEE 802.11 Networks	Remote server	Mobile data offloading in a Wi-Fi network	Resource allocation and task scheduling
Liu et al. 2019 [72]	Dynamic	Partial offloading	Server association, queue length	Multi-Hop	Lyapunov optimization	✓	Mobile edge	✗	Edge-Cloud	Multiple edge server	Reliable edge data offloading and resource utilization	Distributive learning algorithm for reducing latency
Feng et al. 2019 [73]	Static	Binary	Latency, data size, energy consumption	Multi-Hop	Lagrangian dual	✗	Mobile edge	✗	Mobile Edge	Edge Server	Wireless powered data offloading for utilizing resource	Cost defining with this existing model
Wang et al. 2019 [74]	Dynamic	Binary edge Network	Bandwidth, computation rates,	Multi-Hop	Lagreedy algorithm	✗	Roadside Edge units	✗	Edge-cloud	Multi-server	Reliable edge data offloading to reduce latency	Priority based data offloading for heterogeneous networks
Ferrag et al.2019 [75]	Dynamic	Partial offloading	Task execution, uploading rate, transmission rate	Multi-hop	QCQP	✓	Edge network	✓	Edge-Network	Multi-Edge	QCQP for minimizing latency in industrial applications	Priority based offloading and energy optimization
Yang et al. 2019 [76]	Mobile environment	Two-stage	Delay, energy consumption, offloading decision	Multi-Hop	Randomize algorithm	✗	Mobile edge	✗	Mobile edge	Single server	Maritime communication network framework	AI based intelligent decision making policy
Wu et al. 2019 [77]	Distributed	Hierarchical MEC	Bandwidth, workload, task completion	Multi-Hop	BROA	✓	Edge cloud	✗	Edge computing	Mobile edge	Latency aware hierarchical computation offloading	Task migration and resource allocation for distributes networks
Alameddine et al.2019 [78]	Dynamic	Heterogeneity	Latency, server capacity	Multi-hop	DTOS-MIP	✗	Multi-Access Edge device	✗	Multi-access edge	Edge Server	Network-based task offloading in hetero -geneous MCC network	Energy optimization in hierarchical offloading
Zhang et al.2019 [79]	Dynamic	Binary	Task arrival, QoS, processing speed, occupancy	Multi-hop	MASM	✗	Edge device	✓	Cloudlet	VM Edge Server	AI-based delay efficient task offloading	AI based partial computation offloading
Singh et al. 2021 [80]	Dynamic	Total or partial	Latency, Resource	Multi-hop	Lyapunov Optimization	✓	Edge Network	✗	Mobile edge	Edge Server	Server association policy for utilizing resource and minimize delay	AI based task data offloading for increase reliability
Wang et al. 2019 [81]	Distributed	Edge device	Delay, energy, computation	Multi-hop	Distributed algorithm	✗	Edge-cloud	✗	Mobile edge	Edge	M/M/n queueing model for utilizing resources in a distributed environment	Priority based task offloading in wifi networks
Zhu et al. 2019 [82]	Dynamic	Mobile edge	Latency	Multi-hop	POMDP	✗	Edge Device	✗	Vehicular	Cloud	Real-time vehicular analysis	Task interdependency in complex environment

TABLE III
COMPARISON OF EXISTING WORKS WHERE OBJECTIVE IS RESOURCE MANAGEMENT

Contribution	Environment	Types of offloading	Constraints	Distance measure	Proposed technique	Task partition	Execution level	Task priority	System model	Server type	Advantage	Research gap
Xiong et al. 2018 [86]	Dynamic	Distributed,	Delay, service demand	Single hop	Variational inequalities	✗	Edge server	✗	Cloud/Edge	Single server	Blockchain based computation offloading in fog network	Secure and optimal pricing policy for utilizing cloud resources
Chen et al. 2019 [87]	Heterogenus	Distributed	Utility, resource	D2D	Multistage Auction,	✗	Edge server	✗	Wireless network	Multi-server	Auction based resource allocation in heterogenus network	Task priority assignment and auction based resource allocation
Wang et al. 2019 [20]	Dynamic	Distributed	Latency, Power Consumption	Multi-hop	Lyapunov optimization	✗	Edge server	✗	Edge computing	Multiple Edge	Energy efficient task offloading	Resource provisioning
Mukherjee et al. 2019 [45]	Dynamic	Distributed	Monotonicity, IR, IC	Multi-hop	Multi-armed bandit	✗	Edge server	✗	Edge computing,	Multi server	Task offloading and resource sharing in fog network	Cooperation and task migration among servers
Guo et al. 2019 [88]	Wireless Networks	Heterogeneous	Spectrum allocated,	Multi-hop	Reinforcement learning	✗	Edge server	✗	Mobile Edge	Multi server	RL based resource allocation in wireless environment	Adoption of catching schemes with the existing techniques
Chowdhury et al. 2022 [89]	Bilevel	Dynamic	Execution, resource, delay	Multi-hop	Ant colony	✗	Edge server	✗	Mobile edge	Multi server	Multi user cooperative task offloading	Priority based task rescheduling and multiobjective optimization
Zhao et al. 2019 [90]	Vehicular Networks	Dynamic	Computing, resource allocation	Multi-hop	CCORAO	✗	Edge server	✗	Mobile edge	Multi server	Dynamic resource allocation in vehicular network	Intelligent partial computation offloading in ad-hoc networks
Yan et al. 2019 [91]	Wireless Networks	Interdependency	Transmit	Single- hop	Gibbs sampling	✓	Edge server	✗	Mobile-Edge	Multi server	Interdependent data offloading in MEC network	Dynamic frequency for edge servers for large scale IoT data offloading
Kiran et al. 2019 [92]	Wireless Networks	Static	Power, CPU frequency	Multi-hop	Reinforcement learning	✗	Edge server	✗	Mobile edge	Multi server	SDN controller in Wireless environment	Priority based task partitioning and resource provisioning
Hazra et al. 2020 [93]	Hybrid relaying	Partial	Capability,energy computation	Multi-hop	Gibbs ampling	✓	Edge server	✗	Mobile edge	Edge cloud	Efficient computation offloading in HR architecture	Full-duplex network over the existing techniques
Qian et al. 2019 [94]	Cellular Network	Distributed	Computation, transmission power	Single- hop	Nash-stability	✓	Edge server	✗	Mobile edge	Multi-server	Joint communication and computation optimization for IoT	Joint resource allocation and computation offloading in dynamic environment
Dai et al. 2019 [95]	Vehicular network	Partial	Workload, delay,	Single-hop	JSCO	✗	Edge server	✗	Edge Computing	VEC server	Task offloading and load balaning in IoT networks	Unresticted and dynamic vehicular movement in a stochastic environment
Yao at al. 2019 [96]	IoT Networks	Dynamic	Delay, offloads, resource,	Multi-hop	FRPA	✗	Edge device	✗	Edge Computing	Multi VM	Resource provisioning for IoT networks	Edge resource allocation for multiple user
Zhou et al. 2019 [97]	Vehicular network	Dynamic	Cost. QoS, transmission power	Multi-hop	Matching algorithm	✗	Edge device	✗	Vehicular Edge	Multi-server	Matching based resource allocation and pricing matching	Dynamic vehicular movement and channel allocation in a heterogenus enbironment

"✓" Indicates that the research work fully or partly considers this feature.
"✗" Indicates that no consideration is given to the features.

influenced by the spatial separation between IoT devices and computational servers. Each IoT device has some processing capacity for executing small applications. Otherwise, it offloads the applications to computational servers. While executing inside IoT devices, IoT applications require ultralow latency. However, offloading the data to suitable computational servers depends on the distance between the devices. Using edge computing technology may reduce the transmission time of applications [95]. Edge devices are placed near IoT devices, which may minimize latency to zero. In those cases, the computational time of the IoT applications depends on the processing time on the assigned computational server. In contrast, assigning IoT applications to a CDC may require long latency because of the centralized nature of cloud technology. The computational time of applications depends on processing time and latency. Researchers have developed an intelligent offloading strategy to run applications locally or at the edge of the network, reducing overall application latency.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

TABLE IV
EXISTING ENERGY OPTIMIZATION-BASED COMPUTATION OFFLOADING

Contribution	Environment	Types of offloading	Constraints	Distance measure	Proposed technique	Task partition	Execution level	Task priority	System model	Server type	Advantage	Research gap
Kim et al. 2019 [99]	Wireless Network	Partial	Latency, energy	Multi-Hop	JUFO	✗	Edge server	✓	Edge-Cloud	Single server	Edge data offloading in a wireless environment	Energy-Delay tradeoff in heterogeneous environment
Chen et al. 2019 [100]	Static	Deterministic subproblems	CPU, Backlog queue Computation	Multi-Hop	Lyapunov optimization	✗	Edge server	✗	Mobile edge	Edge Server	Energy-efficient task allocation strategy	Individual delay calculation for each task
Sah et al. 2022 [101]	Dynamic	Mobile	Energy	Multi-Hop	Lyapunov drift-plus-penalty	✗	Base stations	✓	D2D	Mobile devices	Energy-efficient and stable data offloading	In a non steady-state environment
Lin et al. 2019 [102]	D2D	Static data	Task completion, causality	One-Hop	Convex optimization	✓	Cooperative Computation	✗	Mobile edge	Edge Server	Time slotted cooperative D2D communication	Tradeoff between energy-Delay optimization
Zhang et al. 2019 [103]	Edge cluster	Fair	Transmission power, Delay	Multi-Hop	FEMTO	✓	Multiple edge server	✓	IoT-edge	Edge Server	Edge-enabled IoT networks	Priority allocation among multiple input tasks
Shan et al. 2019 [104]	Wireless	Static data	Delay, energy, transmission	Two-Hop	Heuristic algorithms	✗	Edge server	✗	Wireless edge	Edge Server	Energy-efficient Heuristic offloading	Multi-connection among large scale computing devices
Wei et al. 2018 [105]	Dynamic	Static data	Latency	Single-Hop	Reinforcement Learning	✗	Edge Processing	✓	Mobile edge	Edge Server	RL based edge data offloading	Large-scale resource allocation for IoT applications
Hazra et al. 2022 [2]	Mobile	Mobile Cloud	Energy , Opportunistic partitioning	Two-Hop	Lyapunov optimization	✓	Cloudlet -Cloud	✗	Mobile Cloud	Single server	Dynamic task offloading decision mechanism	Real-time experimental analysis for heterogeneous networks
Sarkar et al. 2020 [106]	Quasi-static	Partial	Latency, computational speed, resources,	Multi-Hop	Randomize algorithm	✓	Parallel computing	✗	Mobile-Edge	Single server	Mult-iuser Partial computation offloading	AI based adaptive dynamic computation co-offloading
Aazam et al. 2019 [50]	Mobile	Two-stage	Delay, energy, power limitations, decision	Multi-Hop	Randomize algorithm	✗	Mobile edge	✗	Mobile edge	Single server	Maritime communication network framework	Optimal and efficient decision based smart management
Kim at al. 2019 [107]	Static	Two-stage	Service rate, throughput, energy	Multi-Hop	Lyapunov optimization	✗	Cloud VM	✓	Mobile Cloud	Multi VM	Long term energy utilization	Priority based Online task offloading
Deb et al. 2022 [98]	Dynamic	Cloud data	Causality, playback continuity	Single-Hop	Randomize algorithm	✗	Cloud server	✗	Mobile Cloud	Single server	Analysis of real-time video offloading	Intelligent task offloading Strategy in distributed networks
You et al. 2017 [108]	Mobile	Distributed	Latency offloaded bits, transmit power	Multi-Hop	Optimization algorithm	✓	Cloud VM	✗	Mobile-Edge	Multi VM	Shows that energy latency trade-off is not dependent on the Activation of VM	Tradeoff between energy-Delay Optimization
Lagén et al. 2018 [109]	Distributed	Total or partial	Latency	Multi-Hop	Randomize algorithm	✓	Remote VM server	✗	VM allocation	Multi VM server	Energy efficient VM task allocation	Multi user optimal resource allocation

2) Energy Consumption: In addition to consuming energy while processing IoT applications on computational servers, IoT applications transmit data over a reliable network. The transmission energy of IoT applications depends on the distance between IoT devices and computational servers. IoT devices can either run applications locally or offload them to computational servers. IoT applications require no transmission energy while they are running inside IoT devices. However, offloading the data to suitable computational servers depends on the distance. The evolution of edge computing technology could optimize the transmission energy used by applications. Edge devices may reduce transmission power to zero when placed near IoT devices. In those cases, the end-to-end energy consumption of IoT applications depends on the processing power of the assigned computing server [86]. However, transferring IoT applications to a CDC may consume enormous energy due to its centralized nature. The energy consumption rate of IoT applications depends on both transmission energy and processing energy. A brief comparative analysis of the existing works with their advantages and disadvantages is presented in Table IV. Most applications can now be executed locally, on edge nodes, or on edge nodes thanks to several intelligent offloading strategies developed by researchers in recent years. This may minimize the end-to-end energy consumption of applications.

3) Resource Utilization: Resource utilization on a computation server determines how much resources are allocated and utilized for processing IoT applications. These resources, such as CPU, memory, and disk space, are assigned to IoT applications as virtual machine instances or containers. Additionally, IoT devices have a certain resource capacity for processing small IoT applications, resulting in efficient utilization of resources [115]. When tasks are executed on local computation servers, such as edge nodes, the local controller either allocates the server fully for the application or assigns a portion of the server as a virtualization technique, depending on the application requirements. The local computational servers employ lightweight container-based virtualization techniques for delay-sensitive or event-driven applications in order to achieve a faster deployment strategy and enhance parallelism while ensuring optimal utilization of resources. However, some service providers still use VM instances for the virtualization technique to provide more security and privacy for the application with minimal resource utilization compared with the container-centric virtualization technique [116]. A CDC primarily deploys resource-intensive applications using container-driven and virtual machine-based virtualization techniques that assign containers or virtual machines depending on the application's requirements. Increasing parallelism between applications can be achieved by utilizing computing resources effectively. For complex IoT applications, GPUs play an important role in enhancing the performance and efficiency of edge devices. GPUs are also essential for providing the necessary computing power for IoT devices to process data locally. Additionally, GPUs can be used to optimize edge computing applications by offloading tasks that would otherwise be handled by the cloud and supporting ML-based analysis. Table III illustrates a brief overview of the existing works.

4) Load Balancing: Load balancing is essential for computation servers to process IoT applications with minimal waiting and computation time. Load balancing is a mechanism to distribute the load among active servers, which can start up the processing of the application with the shortest possible delay. It is possible to improve the scheduling and application processing of IoT applications with minimal computation delay by inventing various cutting-edge technologies in a distributed environment, such as cloud computing, edge computing, and serverless computing. The system manager helps

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

deploy the IoT applications to a suitable computing environment, efficiently processing the applications and improving the systems' performance while meeting various QoS constraints. For example, serverless computing is mainly ideal for executing event-driven applications. The edge nodes are suitable for delay-sensitive applications, whereas resource-intensive and request-based applications are performed in the cloud. The edge nodes can distribute applications among neighboring nodes and process them independently unless the applications are offloaded to a CDC, resulting in minimal latency. Moreover, computational delay is influenced by the intricacy of the algorithm and the available resource capacity. Effective load balancing of computing servers also maximizes the utilization of computing resources.

5) Context Discovery and Awareness: One of the critical goals of IoT applications running on various computing servers is to find context. Local nodes, such as IoT devices, sensors, actuators, and edge devices, can infer context information, including location, environmental conditions, nearby computer devices, and their present status, among other things [117]. Furthermore, context data is more beneficial for inferring knowledge about the current state of active computing devices in an area. When several IoT users or edge devices with comparable characteristics are nearby, context data can also be used to design efficient and effective data processing policies. A contextually aware and intelligent IoT device and local gateway should be able to automatically decide which IoT devices are best suited to a particular application. This is based on location and capacity. Another context-aware requirement is cooperative and opportunistic sensing. IoT devices should balance their workload with neighboring devices in a location and ensure no resources are wasted [118]. This will increase parallelism among applications and maximize resource utilization. In addition to affecting data collection and fusion strategies, context data plays a crucial role.

6) Mobility: Mobility is a crucial aspect of IoT applications, especially for smart applications like vehicular systems, unmanned aerial vehicles (UAVs), smart grids, and automated transportation. Some IoT applications do not require communication with gateway devices for data analytics and resource discovery, making it unnecessary to deploy expensive gateways at the network edge [119], [120]. With an efficient mobility strategy, IoT gateway devices can be deployed at the network edge to manage numerous distributed IoT devices. In such scenarios, edge devices, such as sink nodes, routers, mobile phones, laptops, and roadside units (RSUs), directly connect with IoT devices to process and manage real-time IoT applications, such as acquiring health status data from ambulances. Additionally, mobility is essential for dynamically discovering resources and configuring them on computational servers. Discovery encompasses security protocols for the movement of each gateway and is crucial for maintaining the reputation of IoT devices [121]. In a distributed environment, edge computing could play an important role in improving mobility between IoT devices and computation servers. Consequently, IoT devices need to connect with gateways for a limited time and transfer data while either the IoT device itself or the corresponding gateway is in motion [122].

Vehicle-to-everything (V2X) communication, for example, enables automobiles to interact with their environment, including other vehicles, roadside infrastructure, pedestrians, and cloud-based applications. The V2X communication system facilitates the exchange of critical information between vehicles in real time, which enhances road safety and optimizes traffic efficiency. Parameters, such as speed, position, and traffic conditions, are included in this information. This communication system is enhanced by including cellular V2X (C-V2X), an extension of conventional V2X technology that utilizes cellular networks to improve reach and reliability. Cellular networks provide extensive geographical coverage, facilitating communication across vast distances and in adverse environmental circumstances. Furthermore, these systems can scale and integrate seamlessly with cloud-based services, enabling the retrieval of up-to-date information and utilizing sophisticated features. The adoption of this global standard significantly enhances the driving experience by improving connectivity and safety. It emphasizes V2X communication, specifically focusing on Cellular V2X, which has become a fundamental aspect of contemporary vehicle technology. A brief summary of existing research is provided in Table V.

7) Security and Privacy: Security and privacy are the two significant objectives for communications between IoT devices, edge servers, and cloud servers [123]. In order to provide accurate information and handle data transactions, security and privacy must be guaranteed. Information and data can be negatively impacted in two ways. First, intruders may disrupt the sensing policy, affecting the network's security and privacy. Second, disruptions to sensing applications and data collection from the environment can negatively affect the transmission of accurate and real-time data to the end users or IoT actuators, impacting system reliability. Security and privacy are also critical issues for modern IoT objects and mainly refer to providing trust in computational servers, including reliable and secure edge and edge devices, potential cyber-attacks on Internet-connected things, authentication and trust, data and network security, and so on [77], [124]. For example, cyberattacks on the IoT devices of a smart city, including edge devices, can affect the overall network's performance and data analysis. Furthermore, end devices, edge servers, and cloud-based data centers cannot deliver accurate services to IoT objects in smart cities, which results in the wrong decisions and reactions to disasters and emergencies. Network security is also an essential aspect of IoT devices for transferring data and information to computational servers without being compromised by sniffer attacks, jamming attacks, etc.

VI. State-of-the-Art Techniques

Edge data offloading is a novel strategy to address the limited battery power of IoT devices by performing certain segments of IoT data on remote processing devices. However, we can achieve greater benefits by incorporating efficient power management and delay-controlled algorithms into the IoT–edge–cloud environment. Offloading decisions are made using AI algorithms and other heuristic or meta-heuristic techniques. For example, queueing theory minimizes task

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

TABLE V
COMPARISON OF EXISTING WORKS WHERE OBJECTIVE IS MOBILITY

Contribution	Environment	Types of offloading	Constraints	Distance measure	Proposed technique	Task partition	Execution level	Task priority	System model	Server type	Advantage	Research gap
Chowdhury et al. 2022 [89]	Static	Parallel	Transmission, arrival rate, processing,	Multi-Hop	Heuristic	✗	Edge RAN	✓	Parallel offloading	AP	A scalable edge data offloading technique	Multi-hop communication in dynamic networking
Singh et al.2021 [80]	Static	Mobile Computation	Time	Multi-Hop	Game theory	✗	Cloud	✗	MCC	Cloud server	Energy-aware shared mobile data offloading	Intelligent decision based task offloading
Cao et al. 2016 [110]	Heterogeneous	HetNets	Network capacity, connection.	Multi-Hop	DATO	✗	Base Station	✗	5G network	D2D	D2D data offloading in heterogeneous cellular Network	D2D communication in a typical and complex data offloading in 5G nwtworks
Mukherjee et al. 2020 [40]	Mobility	Wi-Fi	CPU	Single hop	Comparison	✗	Virtual	✗	Wi-Fi offloading	Single server	Delay based Wi-Fi offloading	Blockage problem in hierarchical edge environment
Chen et al. 2017 [111]	Static	Cache-enabled	Transmission rate, offloading	Multihop	Randomized algorithm	✗	Device-level	✗	D2D communication	Remote server	Energy optimized D2D offloading	Priority based Mobility and security management
Liu et al. 2017 [112]	Heterogeneous	D2D	Connection, spectrum efficiency	Multihop	DAOA	✗	Mobile Device	✗	Radio access networks	Virtual	D2D offloading in Multi-Radio Access	Comparison with existing standard baseline algorithm
Guo et al. 2018 [113]	Distributed	Quasi-static	Latency, transmission rate, offloading decision	Multi-Hop	Game-theoretic	✗	Edge-cloud	✗	Mobile-edge computing	Multi-access Edge	Game-theoretic approach for collaborative data offloading	Collaborative task offloading for heterogenus MEC system
Xie et al. 2019 [114]	Vehicular Networks	Parallel offloading	Delay, energy	Single-hop	Markov model	✗	RSU, edge server	✗	Vehicular edge	Multi-server	Mobility oriented edge data offloading	Energy efficient multihop cooffloading for mobile devices
Misra et al. 2019 [25]	Vehicular Network	Static	Link utilization, processing, energy, mobility, SDN	Multi-Hop	Soft-VAN	✗	Edge device	✗	Vehicular SDN	Centralized	SDN based edge data offloading	Prediction localization in cellular networks
Hazra et al. 2021 [32]	Dynamic	Partial offloading	Cost, latency, failure probability	Multi-Hop	Heuristic Algorithm	✓	Multi-edge server	✗	Mobile Edge	Multi-server	Latency aware mobile data offloading in multiple edge device	Intelligent decision making and bandwidth utilization for D2D communication
Hazra et al. 2018 [2]	Cellular Networks	D2D	Feasibility, IR, IC	Two-Hop	Lagrange multiplier	✓	BS level	✗	Cellular Networks	Single server	Location-aware greedy algorithm	Priority based D2D multicasting for cellular networks
Sarkar et al. 2022 [115]	Radio Access Networks	D2D	Energy, offloading decision	D2D	SWIPT	✗	BBU pool	✗	Radio Access network	RRHs	Joint task offloading in a heterogeneous network	Priority based traffic forwarding for cloud RNA
Aazam et al. 2019 [116]	Distributed	D2D	The data rate, link utilization, transmission order	D2D	MHRC	✗	Edge-Cloud	✗	Edge computing	Single server	Mobility aware caching technique using edge network	Mobility aware obstracle prediction for conplex networks
Adhikari et al. 2019 [117]	Cellular Networks	D2D	Offloading, delay, size transmission power	Multi-hop	Randomized algorithm	✗	Edge cloud	✗	D2D-MEC	Single-server	Cellular based mobile data offloading using D2D communication	Joint resource provisioning and computation offloading orthogonal MEC systems

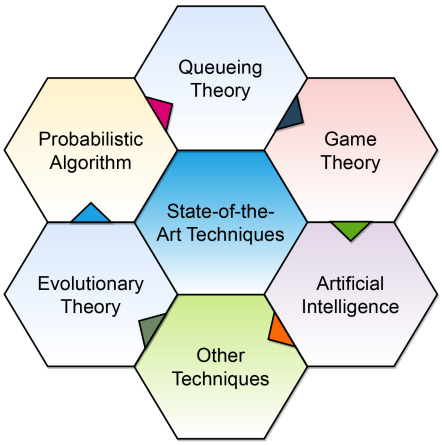

Fig. 10. State-of-the-art techniques for optimizing IoT objectives.

waiting times, and statistical models are employed to predict network performance. It is also essential to note that other state-of-the-art methods described in existing literature are widely used for optimizing IoT objectives in the future [26]. These techniques encompass probability theory, evolutionary theory, stochastic optimization techniques, and more. Fig. 10 illustrates some of the important methods used for optimizing IoT objectives.

A. Queueing Theory

Queuing theory is an area of mathematics investigating how lines/queues form, perform, and break down. Using queueing theory, the system can examine the arrival tasks, service rate, number of servers, system spaces, and IoT devices. Many practical applications use queueing theory to maintain traffic flow, congestion, communication, and facility design in shopping malls, post offices, and banks. The M/M/1 and M/M/c queues are widely used in the field of queuing theory. The M/M/1 queue assumes that service times follow an exponential distribution, while the M/M/c queue is designed to handle scenarios with numerous servers. M/G/1 queue models include broad service time distributions, while M/D/1 queue models address deterministic service time distributions. Priority queues are used to assign priority to consumers based on their importance, whereas multiclass queues are used for handling a variety of customer types. Bulk queues are designed to manage the arrival of groups, whereas finite queues take into account capacity constraints.

Queueing models follow a variety of disciplines, including first come–first served (FCFS), last in–first out (LIFO), and shortest job first (SJF). It aims to provide stability that is both effective and affordable. Several research initiatives have been done to incorporate the advantages of queueing theory for computation offloading. For example, Hazra et al. [90] have designed an energy-effective computation offloading framework using queueing theory in the edge computing environment. The authors claim that this framework controls the task offloading rate and optimizes energy while achieving the least possible delay. Li et al. [125] have designed a resource allocation and computation distribution strategy for heterogeneous edge networks. The authors combined the Lyapunov function and queueing theory to optimize the response time of real-time IoT applications. In some cases, the queueing system suffers from high waiting times due to the frequent occurrence of IoT tasks.

B. Game Theory

Game theory describes how self-interested actors interact dynamically. The game theory encompasses frameworks for modeling complicated interactions and alternative solutions to describe an election's logical outcome. The aim of game theory is to develop alternative strategies for competing against each other. This strategy is also vital for responding to changes

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

in relevant topics through decision making. Additionally, game theory can be used to determine centrality, discover communities, forecast linkages, distribute resources, assign tasks, manage privacy and security techniques, and collect data, among other things. Academics and industries are likely to benefit significantly from applying game theory to IoT technology. Although various game-theoretic approaches exist in the literature, cooperative and noncooperative games are popular strategies often used in IoT data offloading scenarios. Several research efforts have been made on game theory to negotiate processing costs, optimize offloading prices, and minimize jamming attacks. Hazra et al. [35] have developed a cooperative game theory-based system to optimize delay in edge networks. Similarly, Mukherjee et al. [41] have also presented a deadline-based scheduling strategy to maximize service providers' revenue in an edge federation environment.

C. Artificial Intelligence

AI is becoming a prevalent tool in many application domains, including IoT, cloud computing, production, management, and industrial operations. Over the last two years, academics and industries have been trying to combine the benefits of AI and IoT [126]. Integrated AI solutions, such as ML-based analytics and graph-based pathfinding problems, are already applied by major IoT platforms and service providers. With ML applications for IoT, businesses can minimize unplanned downtime, increase operational efficiency, develop new products and services, and manage risk. While IoT gives us data, ML helps us unlock answers, enabling intelligent actions by providing context and creativity. With the help of ML, businesses can evaluate the data provided by sensors and make informed decisions based on that information. With supervised learning, IoT tools can predict equipment failures and optimize maintenance schedules based on historical data. For security and fault detection, it is also used to classify data as normal or abnormal based on labeled data sets. Using unsupervised learning in IoT is beneficial for clustering and pattern recognition, especially for analyzing sensor data. Without labeled data, it helps detect anomalies and uncover hidden structures. Large-scale IoT data must be organized, resource allocation must be optimized, and irregularities must be identified in complex networks.

On the other hand, deep learning-based algorithms are widely adopted in many application domains, such as industrial applications, transportation systems, agriculture, etc. Deep learning-based applications are also used in edge networks to optimize delay, cost, resources, and energy consumption. In some cases, deep learning techniques are also utilized to make optimal decisions about selecting a suitable computing device. Deep neural networks analyze large volumes of user data, and RNN/LSTM-derived models are applied to predict time-series data. Techniques based on RL are widely used to control the uncertainty of the network. Deep reinforcement learning (DRL)-based algorithms manage and control ample state and action space in an unknown environment [127]. In IoT, deep Q-network (DQN) and actuator-to-gateway (A2G) enable efficient decision making and control. With DQN, intelligent agents can make better decisions based on reinforcement learning, while with A2G, IoT devices, and the central gateway can communicate and be controlled more effectively. In the realm of IoT, a fascinating application of DRL emerges when IoT devices are cast as *agents*. Within this framework, the offloading strategies become the *actions* these intelligent agents take. The objective function value then transforms into the *reward* as a measure of success. IoT systems can autonomously adjust and optimize resource allocation using this approach, opening up new opportunities for efficient and dynamic decision making. The analysis of the DRL strategy in computation offloading is also presented in Fig. 11 while considering the system model and experimental setup given in [37]. Several experiments and existing literature also validate the applicability of deep learning techniques compared to standard baseline algorithms. If we look carefully, we can observe that deep learning techniques can quick self-learning and self-adaptive complex network functionalities. Due to these advantages, deep learning techniques are also widely used for making computation offloading decisions over edge networks, irrespective of the types of edge networks and their functionalities.

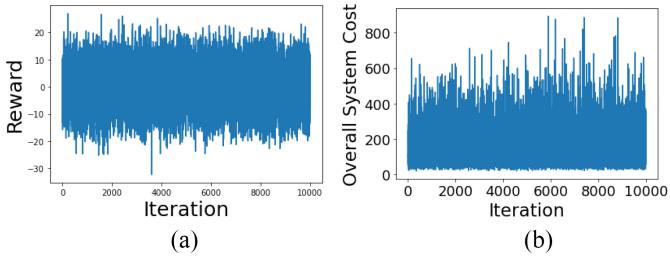

Fig. 11. Performance analysis of edge networks for offloading. (a) System reward. (b) Total execution cost.

D. Evolutionary Theory

Evolutionary computing approaches are adaptable and resilient, making them ideal for dealing with nonlinear, high-dimensional, and complex engineering problems. These strategies do not need any specific scenario, structure, or differentiable database to provide near-optimal solutions to issues that were not well-defined previously. These approaches enable the expert to present numerous near-optimal solutions very efficiently. Since the 1950s, many evolutionary algorithms have been proposed, including genetic algorithms, ant colony optimization, particle swarm optimization, etc. Specifically, evolutionary computing is the set of globally efficient algorithms inspired by biological evolution. This computing algorithm has characteristics that change with each generation. The number of times the algorithm is applied, and each time it delivers a more optimal output until the target saturation level is reached. This is referred to as a generation. In edge computing, evolutionary computing technology is used to optimize data processing while considering multiple networks and system-level QoS limitations. Other research activities may be found in [114] and [128], where authors use evolutionary algorithms to solve scheduling, offloading,

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

and load balancing problems. The key objective of applying evolutionary algorithms is to adopt complex network functions and provide nature-inspired solutions.

The advantages of evolutionary computing in edge computing include optimizing edge resources, adapting to dynamic environments, reducing energy consumption, and providing fault tolerance. As an example, evolutionary computing can be used to optimize processing power, storage space, and communication bandwidth at the edge. These resources allow edge computing systems to work more efficiently and effectively. Additionally, evolutionary computing makes it easier for edge computing systems to adapt to changing environments and workloads. Evolutionary computing can continue to run at its best when conditions change by optimizing system parameters constantly. Evolutionary computing can also be used to reduce edge computing energy consumption. The end-to-end energy consumption of the edge computing system can be reduced if the amount of energy needed to perform a given task is kept to a minimum. This is especially critical in places with few resources. On the other hand, evolutionary computing can increase edge computing fault tolerance and ensure the system runs despite failures. It is possible by ensuring resources are distributed and utilized efficiently. The exact benefits of evolutionary theory in edge computing may be difficult to achieve due to complications like hyperparameter tuning, approximate results and biased decisions, and randomness. Despite this, there is still much room for improvement, and many initiatives have already been undertaken.

E. Probabilistic Algorithm

Probability theory is extremely significant to almost all branches of mathematics since it provides mathematical explanations of how likely it is for something to happen or to be true. Probability and statistics are inextricably interlaced because statistical data is regularly studied to see if reasonable inferences can be formed about a particular incident and make predictions about future events [129]. In the IoT domain, probabilities estimate energy wasted by IoT devices. They are also used to predict network congestion, predict subsequent downtime, and make task offloading decisions by selecting the most suitable devices. For example, Liao et al. [130] have developed a computation offloading strategy for massive IoT applications. Further, the authors applied a probabilistic technique to analyze the tradeoff between energy and delay. Hazra et al. [31] have designed an edge device selection strategy for optimizing energy–delay cost in a federated edge environment. Similarly, Wang et al. [131] have also developed a probability-preferred offloading strategy for delay-sensitive IoT applications in MEC networks.

F. Other Techniques

Linear, nonlinear, and dynamic programming are essential tools in optimization, owing to their versatility and extensive range of applications [132]. For example, linear programming is crucial for solving problems characterized by linear relationships, and it is widely used in several application domains, such as supply chain management, logistics, and resource allocation. A nonlinear programming approach broadens its use by addressing nonlinear interactions, thus making it well-suited to complex systems used in engineering and economics. On the other hand, dynamic programming approaches are particularly useful when there are overlapping subproblems and optimal substructures in a problem. Through this process, it is possible to develop efficient solutions to decision-making problems that occur over a period of time over the course of a period of time. IoT systems at the edge can be improved by efficiently allocating and managing resources, energy, and decision-making processes associated with these strategies in the context of IoT networks at the edge.

VII. Smart Healthcare: Use Case

This section discusses a more relevant use case study in the IoT–edge–cloud environment. Many people worldwide suffer from critical diseases, such as cardiovascular and diabetes. These diseases, such as kidney failure, heart attacks, irregular heart rhythms, and strokes, directly affect human health. A delay in action or incorrect treatment may endanger patient health. Therefore, continuously monitoring patient health using a real-time environment may solve such an issue [133]. The advancement of IoT devices with the involvement of many advanced technologies, including wireless body area networks, wireless sensor networks, and wearable sensors, can resolve hospital patient monitoring systems' current challenges and difficulties. Wearable and wireless sensors should collect biosignals from the human body and transmit them to a local computing server to measure the patient's health [2]. These sensors have limited resource capacity and transmit the sensed data efficiently to computational servers for processing, analysis, and taking actions with minimal energy consumption.

To enhance healthcare services and the patient monitoring system with minimal latency, computing devices should be distributed locally and placed near IoT sensors. Edge computing supports such architectures, and distributed edge devices can effectively process real-time patient health data with minimal latency. These functionalities may improve the real-time data analysis capabilities of edge devices [134]. In contrast, the centralized cloud server will take the longest to analyze for taking specific action by hospital authorities such as doctors or nurses for a critical patient due to long-distance data travel and any resulting delays. The IoT devices can offload data from the nearest edge devices to the nearest edge devices, which has the advantages of edge computing, real-time patient monitoring data, and healthcare data [135]. This strategy helps to minimize latency for real-time data and reduces the end-to-end energy consumption rate and processing cost of computing resources.

The three-layer edge patient monitoring control architecture is shown in Fig. 12, where eight key essential steps are considered to illustrate the overall IoT-based patient monitoring system. In step one, health-related information of the patient should be recorded from the body or IoT-implemented sensors connected to the patient's body, including multiple parameters of the patient monitoring system. In step two, the sensed data should be offloaded to remote computing devices

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

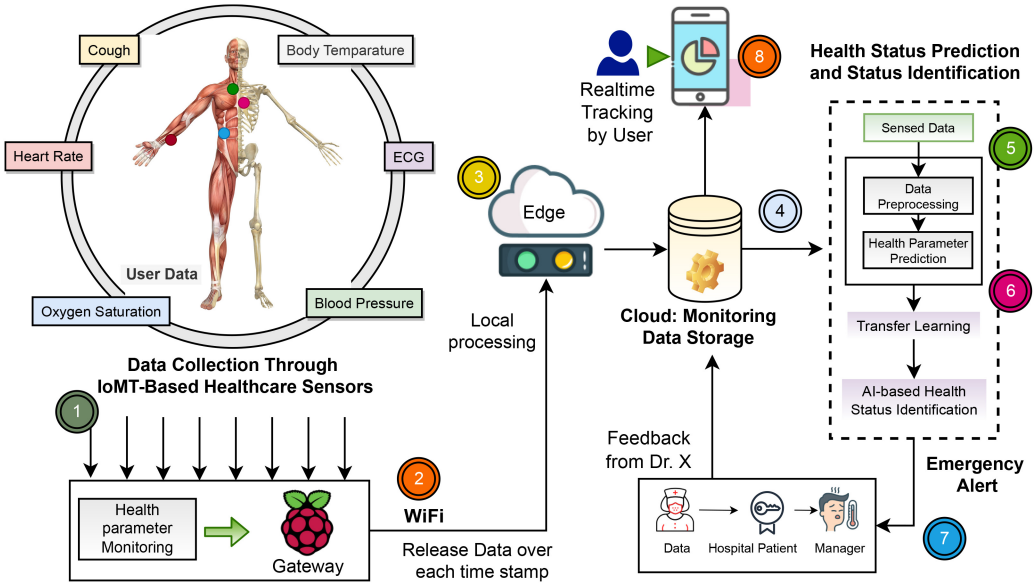

Fig. 12. Illustration of an IoT-based patient monitoring system.

through smart IoT gateway devices for further processing and analysis, including context-aware information, such as time, blood pressure, heartbeat, and temperature. Context-aware data is helpful in identifying unusual patterns and generating valuable health-related information about the patient. The small and cost-effective wearable devices are mainly used to collect health information about a patient and wirelessly (using WiFi, Bluetooth, and ZigBee) transmit high-resolution signals (i.e., respiration rate and ECG) to nearby edge nodes through some IoT gateways [136].

In step three, distributed edge devices provide advanced services, including real-time data processing, real-time decision making, and temporary data storage. During regular operation, the edge devices retrieve data from IoT wearable devices and monitor the patient's health status. Real-time actuators, such as alarms and edge devices, alert health administrators, doctors, and nurses about abnormal conditions of patients. For resource-intensive applications such as medicine, classification based on human health should be analyzed and processed on cloud servers, as illustrated in step four. edge devices also offload data to cloud storage via the cloud gateway and MQTT protocol. The cloud server analysis also contains four steps. Following data preprocessing, the cleaned and preprocessed data is used for disease analysis, where features are extracted [137]. AI-based algorithms, such as ML, DL, and federated learning-based techniques, are used to predict the patient's condition. These techniques are used to ensure that the AI-based algorithms are able to accurately identify patterns and diagnose diseases in inpatient data while preserving and securing users' data. The edge device can free up its resources and focus on disease analysis by offloading data to the cloud storage. The preprocessed data is also necessary for the AI-based algorithms to accurately identify patterns and diagnose diseases. Further, AI-driven techniques are employed to identify the current status of the patient's health on a time-to-time basis. Doctors can prescribe

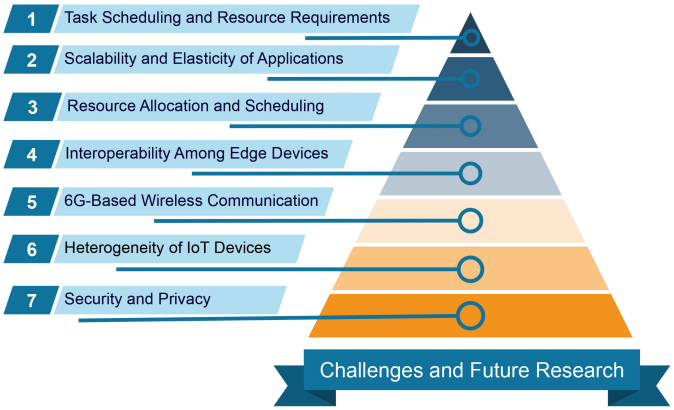

Fig. 13. Edge execution challenges for IoT data.

appropriate medication and tests for patients based on the analysis. These steps are also illustrated in steps 5–7 in Fig. 12. Finally, patients collect their health information, visualize prescribed information and monitor their health status through a mobile application. The patient can also communicate with doctors and discuss their requirements if required [138]. These functionalities present the clean picture and additional benefits of implementing edge computing systems in healthcare applications.

VIII. Potential Challenges and Future Research Directions

As edge computing becomes a more common computing paradigm, it is increasingly used in next-generation IoT applications [139]. However, edge computing presents numerous research problems that need to be addressed in the immediate future. These issues include workload sharing, interoperable data transmission, dynamic pricing, security, reliability, etc., as illustrated in Fig. 13. The following sections highlight

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

several evolving challenges (or opportunities) of computation offloading in an edge computing environment for large-scale IoT deployment.

A. Task Scheduling and Resource Requirements

The primary intent of next-generation IoT technologies is to enhance the capabilities of the IoT ecosystem by incorporating communication protocols, networking technologies, and high computing resources efficiently. Whenever service providers receive sensed data, including numerous signals, noisy data, and time-critical data, each with a different format, response time, size, and resource requirements, they must process data with efficient scheduling techniques. Also, service providers prioritize resource demands, as a lack of resource utilization techniques increases queue waiting time while receiving or executing IoT requests. As a result, the primary goal of task scheduling with diminishing resource requirements is to identify an ideal computing device with sufficient resource capacity that can satisfy task deadlines while maximizing resource efficiency [37]. For example, patient monitoring data should be transferred to edge devices for faster data processing while staying under network restrictions.

Although the challenges lie in the capacity of edge devices. In most scenarios, the binary task offloading process fails due to inefficient priority selection techniques. Besides partial offloading, partial co-offloading and edge federation model help combine available edge and cloud resources into one place. Co-offloading partially facilitates interedge scheduling before edge device execution. However, several efforts have been made to establish a reliable edge network for IoT applications. For example, Hazra et al. [31] proposed an AI-enabled partial service provisioning strategy for industrial IoT Networks. Similarly, Mukherjee et al. [44] also designed a partial computation offloading scheme for delay-sensitive IoT applications. Still, several other challenges related to compatibility and portability among edge devices need to be solved. Additionally, edge network security, privacy, and scalability need to be considered to provide a reliable and secure network for IoT applications, which is still an open challenge to the research community.

B. Scalability and Elasticity of Applications

Edge-enabled IoT data offloading differs from existing distributed computing models like cloud offloading and cloudlet offloading, where resources are unlimited and requests are unrestricted. Two apparent drawbacks can be observed in edge-enabled IoT offloading strategies [4]. First, unpredictable service demand for computing resources may be restricted due to limited processing and storage capability, which reduces users' Quality-of-Experience (QoE) behavior. As service demand increases, the service level agreement may also be affected, or a significant investment may be needed to meet users' QoS expectations [140]. The first QoS advantage derives from recognizing flexible network services, whereas external processing provides a clear and significant improvement over the current system. Second, elasticity is guaranteed if algorithms can be executed automatically based on resource demand and utilize the available edge resources. To ensure that QoS expectations will be met, it is often necessary to make changes to existing services or add new services. This may require a significant financial investment to purchase the necessary resources or upgrade existing ones. Moreover, algorithms must be developed to adjust resources based on demand. This will ensure that the service level agreement is maintained, and users' QoS expectations are satisfied.

The challenges related to scalability and elasticity can be divided into architecture-level and algorithmic-level research issues. Edge devices must be incorporated with parallel processing technologies, such as multicore processing, cluster processing, and advanced computing devices (e.g., GPU). They also must provide on-demand services to users combined with such heterogeneous architectures. For sequential algorithms to be able to address the limitations accrued by dynamic algorithms, new algorithms need to be developed [141]. Such encouragement encourages AI, RL, evolutionary algorithms, optimization techniques, and stability algorithms. An edge architecture should also minimize power consumption by incorporating scalable hardware and algorithms. Using solid-state drives and dynamic random access memory increases scalability and memory access frequency while reducing power consumption and cost [142]. Furthermore, elasticity can be increased by applying an accurate prediction algorithm, which will help manage edge resources dynamically. Edge computing also requires an efficient communication mechanism between edge devices and the cloud. For instance, edge devices should support low-latency, low-bandwidth communication using 5G and edge-to-cloud networks. Additionally, edge devices should be equipped with AI capabilities for real-time analysis, inference, and decision making.

C. Resource Allocation and Scheduling

Current edge devices (e.g., IoT gateway, edge server, and edge router) have limited storage and processing capabilities, limited to a few hundred over the network [143]. In order to achieve high usability and customer satisfaction, it is necessary to implement appropriate resource provisioning and scheduling strategies efficiently for IoT applications. Edge resources have been utilized under random resource allocation or static resource provisioning techniques, which assign a fixed amount of edge resources to devices as needed [113]. Thus, there is a need for VM provisioning, VM migration, and dynamic load-balancing strategies for edge-level devices. A wide range of research has been developed in resource management and planning over time, although various challenging issues and limitations remain open. For example, sharing edge resources among the requesting devices is an NP-hard problem, like some resources are underutilized or some resources are frequently accessed.

Furthermore, methods of estimating demand and forecasting workload can be unreliable and pose an interesting question: *whether AI or ML-based techniques will be helpful for future resource estimation and resource utilization?* Another challenge is the provision of priority-based resource allocation during the designated load day time, i.e., *can*

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

hybrid edge or collaborative edge workload sharing handle various time-restricted delay-sensitive IoT applications? Risks correlated with autoscaling, VM migration-related security, and management challenges are inadequately addressed [134]. Therefore, how to allocate the exact amount of resources and how much resources to allocate simultaneously so that edge resources can be optimally utilized for a long time remains a challenge. This challenge can be addressed by using predictive AI-based models which can accurately estimate the demand for resources and allocate them accordingly. AI-based models can also help identify patterns in usage and predict when resources might be needed. This will enable organizations to better manage edge resources and ensure they are used optimally.

D. Interoperability Among Edge Devices

Interoperability is the ability of smart devices to communicate, connect, and understand information from each other. Interoperability between heterogeneous devices and services is difficult as organizations and service providers work toward completing their portfolios. Edge device interoperability issues can be categorized into three levels: 1) interconnection interoperability; 2) compatibility; and 3) portability. The first challenge lies in interconnection and interoperability among edge devices, where minimal techniques and approaches are available to aggregate multiple edge functionalities and applications. The second challenge is the portability of data and applications in IoT–edge–cloud architecture. These portability issues result from moving data and applications from one edge device to another [144]. To overcome these issues, interoperability standards must be developed and implemented. Additionally, existing edge–cloud solutions must be improved and optimized for data and application portability.

As a result, interoperability challenges extend beyond functional interfaces within an application or service. Security and business models also need to be considered. Very few application programming interfaces (APIs) are available for edge data offloading and execution, which means switching from one edge device to another with comfortable application functionality. In addition, a mapping layer or isolation between their applications and edge applications is needed. New technologies like fog-bus, 5G or beyond 5G-based wireless technologies, and blockchain-based security mechanisms can be incorporated into the isolation layer [145]. Another area of interest is the engagement of brokers, which can provide consumers with maps and standardized application interface options. In portability, application portability is much more significant than data portability. There should be some standardization in the case of portability issues, such as standardized interfaces, standard communication protocols, standard device configurations, and standard service provisioning, which still need to be incorporated and made open for development. Additionally, adopting an open-source platform and container technology (e.g., Docker) for the independent deployment of IoT applications has also encouraged researchers to delve into this particular domain.

E. 6G-Based Wireless Communication

Existing wireless technologies, including Bluetooth, WiFi, 5G, and beyond 5G, have various limitations due to data handling capability on edge networks, necessitating a next-generation wireless technology called 6G communication. For edge networks, 6G technology allows for high mobility, low energy consumption, and low latency despite handling large amounts of user data. According to the 5G analysis and deployment report, 6G technology can address the shortcomings of 5G coupled edge networks, such as rural area coverage, connecting Internet of everything objects, providing satisfactory performance for industrial applications, and meeting the requirements and goals of users [145]. However, 6G technology will require enhanced processing technologies, portable communication protocols, state-of-the-art hardware equipment, multiantenna technology, and adaptive AI for full deployment and coverage. In contrast to 5G and B5G, 6G technology has the potential to transform data transmission at edge networks from connected things to connected intelligence by meeting the needs of real-time applications [113]. However, this technology is not yet commercially available. There are still many 6G-related challenges to be addressed, including quantum ML for 6G communication, the ultrareliable 6G network based on edge intelligence, beam space multiplexing, high security, and privacy. Furthermore, 6G must deliver ultrahigh data rates and ultralow latency while also being energy-efficient and cost-effective. Achieving these goals demands substantial investment and research collaboration from both industry and academia.

F. Heterogeneity of IoT Devices

The heterogeneity issues related to edge computing can be categorized into IoT, middleware, and edge-level heterogeneity [146]. Prior to the implementation of edge-level processing, workload management and memory organization pose challenges. At this level, VM allocation, container deployment, and integration of cutting-edge hardware technology (such as CPU, GPU, and FPGA) in a heterogeneous environment are the current research focus. However, these approaches do not consider the QoS requirements of IoT applications, such as latency, bandwidth, and packet loss, which are essential for latency-critical applications such as video streaming. Therefore, developing algorithms that dynamically allocate resources based on the current network and energy-related parameters and QoS requirements is important.

On the other hand, many hardware companies have internal hardware and software architectures that serve only their organizational needs. Though there are several low-level programming languages, such as OpenCL and CUDA, the disparity between the programming language and advanced hardware makes implementing advanced processing units for edge devices problematic. This indicates that the open challenges are managing resources based on dynamic allocation using VM, workload management among heterogeneous devices, and deploying hardware-compatible programming languages, which can significantly improve edge data offloading performance. These challenges indicate a new edge

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

federation platform that combines all the underlying heterogeneity issues related to offloading. Solutions should also provide a unified interface to manage edge resources and enable secure and resilient communication between edge nodes. It is also essential to keep in mind that the requirement for interoperability and standards for devices and their usage on the IoT cannot be ignored.

G. Security and Privacy

In edge computing, processing numerous IoT applications presents significant security and privacy challenges compared to centralized cloud technology. Security attacks, such as session hijacking, script injection, cross-site channels, and SQL injection are commonly used to compromise data privacy and security on distributed edge devices [116]. These issues primarily stem from long-distance or multihop data transfers from IoT devices to destination computing devices. Additionally, local edge devices face numerous challenges due to the lack of authentication mechanisms and increased potential threats, such as denial-of-service and man-in-the-middle attacks. Researchers have begun deploying lightweight container technology into edge networks to address these challenges in handling time-critical IoT applications. Despite containers and computing resources sharing the same kernel, any network vulnerabilities must be addressed, and system-level authentication mechanisms should be minimized to enhance edge-level security [147]. Containers also face security concerns when processing IoT applications, including compromised secrets, denial-of-service attacks, poisoned images, and container breakouts. Multitenancy in edge environments can undermine network security and privacy. Given the limited computational capability among edge devices, neither developers nor researchers should employ public-key cryptography throughout the edge environment. As a result, future research should focus on ensuring the security and privacy of IoT applications in a standard edge computing environment. Further, improvements in privacy-preserving ML techniques can help secure healthcare data by enabling broader sharing and research of patient data.

IX. Conclusion

Over the past several years, edge computing frameworks have revolutionized the future generation of IoT applications and propelled computing technologies to new heights. Although this technology has numerous limitations and challenges, it has the capability to directly support latency-critical IoT applications. Along with this, the demand for edge-enabled computing paradigms and offloading technologies is increasing dramatically. This survey paper discusses the requirements for edge offloading, offloading techniques, state-of-the-art challenges, and future research opportunities in data offloading to reduce energy demand, processing delay, cost, mobility, and resource management-related issues. This article will motivate researchers and provide clear direction on how to address these challenges with edge computing technology. In the near future, we will adopt several interoperable communication and offloading technologies to provide seamless communications between devices for Industry 5.0 applications.

TABLE VI
Important Abbreviations

Symbols	Definition
3G	Third Generation
5G	Fifth Generation
AI	Artificial intelligence
AP	Access Point
BS	Base Station
BS	Base Station
CPU	Central Processing Unit
D2D	Device-to-Device
FPGA	Field Programmable Gate Array
FTP	File Transfer Protocol
GPU	Graphics Processing Unit
IIoT	Industrial Internet of Things
IoT	Internet of Things
LTE	Long Term Evolution
LTE	Long-Term Evolution
MCC	Mobile Cloud Computing
MEC	Mobile Edge Computing
ML	Machine Learning
QCQP	Quadratically Constrained Quadratic Program
QoE	Quality of Experience
QoS	Quality of Service
RAN	Radio Access Network
RL	Reinforcement Learning
RSU	Road Side Unit
SSD	Solid-State Drive
UMTS	Universal Mobile Telecommunications System
VM	Virtual Machine
WiFi	Wireless Fidelity
WPT	Wireless Power Transfer
XML	Extensible Markup Language

Appendix

The abbreviations are listed in Table VI.

References

[1] M. Aazam, S. Zeadally, and E. F. Flushing, "Task offloading in edge computing for machine learning-based smart healthcare," *Comput. Netw.*, vol. 191, May 2021, Art. no. 108019.

[2] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Fog computing for energy-efficient data offloading of IoT applications in industrial sensor networks," *IEEE Sensors J.*, vol. 22, no. 9, pp. 8663–8671, May 2022.

[3] A. Hazra, P. K. Donta, T. Amgoth, and S. Dustdar, "Cooperative transmission scheduling and computation offloading with collaboration of fog and cloud for industrial IoT applications," *IEEE Internet Things J.*, vol. 10, no. 5, pp. 3944–3953, Mar. 2023.

[4] R. Buyya et al., "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Comput. Surveys*, vol. 51, no. 5, pp. 1–38, 2018.

[5] S. Dustdar, C. Avasalcai, and I. Murturi, "Invited paper: Edge and fog computing: Vision and research challenges," in *Proc. IEEE Int. Conf. Service Orient. Syst. Eng. (SOSE)*, 2019, pp. 96–9609.

[6] M. Mukherjee, L. Shu, and D. Wang, "Survey of fog computing: Fundamental, network applications, and research challenges," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 3, pp. 1826–1857, 3rd Quart., 2018.

[7] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile cloud Comput.*, 2012, pp. 13–16.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

[8] N. Kumari, A. Yadav, and P. K. Jana, "Task offloading in fog computing: A survey of algorithms and optimization techniques," *Comput. Netw.*, vol. 214, Sep. 2022, Art. no. 109137.

[9] A. Kaswan, P. K. Jana, and S. K. Das, "A survey on mobile charging techniques in wireless rechargeable sensor networks," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 3, pp. 1750–1779, 3rd Quart., 2022.

[10] A. Shakarami, M. Ghobaei-Arani, and A. Shahidinejad, "A survey on the computation offloading approaches in mobile edge computing: A machine learning-based perspective," *Comput. Netw.*, vol. 182, Dec. 2020, Art. no. 107496.

[11] A. Shakarami, M. Ghobaei-Arani, M. Masdari, and M. Hosseinzadeh, "A survey on the computation offloading approaches in mobile edge/cloud computing environment: A stochastic-based perspective," *J. Grid Comput.*, vol. 18, pp. 639–671, Nov. 2020.

[12] S. Yuan, Y. Fan, and Y. Cai, "A survey on computation offloading for vehicular edge computing," in *Proc. 7th Int. Conf. Inf. Technol. IoT Smart City*, 2019, pp. 107–112.

[13] M. Aazam, S. Zeadally, and K. A. Harras, "Offloading in fog computing for IoT: Review, enabling technologies, and research opportunities," *Future Gener. Comput. Syst.*, vol. 87, pp. 278–289, Oct. 2018.

[14] M. Mukherjee, M. Guo, J. Lloret, and Q. Zhang, "Leveraging intelligent computation offloading with fog/edge computing for tactile Internet: Advantages and limitations," *IEEE Netw.*, vol. 34, no. 5, pp. 322–329, Sep./Oct. 2020.

[15] H. Flores, P. Hui, S. Tarkoma, Y. Li, S. Srirama, and R. Buyya, "Mobile code offloading: From concept to practice and beyond," *IEEE Commun. Mag.*, vol. 53, no. 3, pp. 80–88, Mar. 2015.

[16] M. Aazam, S. Zeadally, and K. A. Harras, "Fog computing architecture, evaluation, and future research directions," *IEEE Commun. Mag.*, vol. 56, no. 5, pp. 46–52, May 2018.

[17] R. N. B. Rais and O. Khalid, "The insights of mobile data offloading: A comparative study," in *Proc. IEEE 11th Int. Conf. Ubiquitous Future Netw. (ICUFN)*, 2019, pp. 315–320.

[18] A. Boukerche, S. Guan, and R. E. D. Grande, "Sustainable offloading in mobile cloud computing: Algorithmic design and implementation," *ACM Comput. Surveys*, vol. 52, no. 1, p. 11, 2019.

[19] J. Wang, J. Pan, F. Esposito, P. Calyam, Z. Yang, and P. Mohapatra, "Edge cloud offloading algorithms: Issues, methods, and perspectives," *ACM Comput. Surveys*, vol. 52, no. 1, p. 2, 2019.

[20] C. Jiang, X. Cheng, H. Gao, X. Zhou, and J. Wan, "Toward computation offloading in edge computing: A survey," *IEEE Access*, vol. 7, pp. 131543–131558, 2019.

[21] H. Zhou, H. Wang, X. Chen, X. Li, and S. Xu, "Data offloading techniques through vehicular ad hoc networks: A survey," *IEEE Access*, vol. 6, pp. 65250–65259, 2018.

[22] P. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4228–4241, Oct. 2020.

[23] H. Wu, "Multi-objective decision-making for mobile cloud offloading: A survey," *IEEE Access*, vol. 6, pp. 3962–3976, 2018.

[24] A. A. Sadri, A. M. Rahmani, M. Saberikamarposhti, and M. Hosseinzadeh, "Data reduction in fog computing and Internet of Things: A systematic literature survey," *Internet Things*, vol. 20, Nov. 2022, Art. no. 100629.

[25] L. Kong et al., "Edge-computing-driven Internet of Things: A survey," *ACM Comput. Surveys*, vol. 55, no. 8, pp. 1–41, 2022.

[26] A. Hazra, P. Rana, M. Adhikari, and T. Amgoth, "Fog computing for next-generation Internet of Things: Fundamental, state-of-the-art and research challenges," *Comput. Sci. Rev.*, vol. 48, May 2023, Art. no. 100549.

[27] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "A collaborative computational offloading strategy for latency-sensitive applications in fog networks," *IEEE Internet Things J.*, vol. 9, no. 6, pp. 4565–4572, Mar. 2022.

[28] K. Peng et al., "Joint optimization of service chain caching and task offloading in mobile edge computing," *Appl. Soft Comput.*, vol. 103, May 2021, Art. no. 107142.

[29] M. A. A.-A. Mostafa and A. Khater, "Horizontal offloading mechanism for IoT application in fog computing using Microservices case study: Traffic management system," in *Proc. IEEE Jordan Int. Joint Conf. Elect. Eng. Inf. Technol. (JEEIT)*, 2019, pp. 640–647. [Online]. Available: https://api.semanticscholar.org/CorpusID:159042482

[30] K. Akutsu, T. Phung-Duc, Y.-C. Lai, and Y.-D. Lin, "Analyzing vertical and horizontal offloading in federated cloud and edge computing systems," *Telecommun. Syst.*, vol. 79, pp. 447–459, Jan. 2022.

[31] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Collaborative AI-enabled intelligent partial service provisioning in green industrial fog networks," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 2913–2921, Feb. 2023.

[32] A. Droob et al., "Fault tolerant horizontal computation offloading," in *Proc. IEEE Int. Conf. Edge Comput. Commun. (EDGE)*, 2023, pp. 177–182.

[33] I. Sarkar, M. Adhikari, N. Kumar, and S. Kumar, "Dynamic task placement for deadline-aware IoT applications in federated fog networks," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1469–1478, Jan. 2022.

[34] B. Chakraborty, D. M. Divakaran, I. Nevat, G. W. Peters, and M. Gurusamy, "Cost-aware feature selection for IoT device classification," *IEEE Internet Things J.*, vol. 8, no. 14, pp. 11052–11064, Jul. 2021.

[35] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Stackelberg game for service deployment of IoT-enabled applications in 6G-aware fog networks," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5185–5193, Apr. 2021.

[36] A. Hazra and T. Amgoth, "CeCO: Cost-efficient computation offloading of IoT applications in green industrial fog networks," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6255–6263, Sep. 2022.

[37] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Intelligent service deployment policy for next-generation industrial edge networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 9, no. 5, pp. 3057–3066, Sep./Oct. 2022.

[38] Y. Song, F. R. Yu, L. Zhou, X. Yang, and Z. He, "Applications of the Internet of Things (IoT) in smart logistics: A comprehensive survey," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4250–4274, Mar. 2021.

[39] M. Mukherjee, V. Kumar, J. Lloret, and Q. Zhang, "Revenue maximization in delay-aware computation offloading among service providers with fog federation," *IEEE Commun. Lett.*, vol. 24, no. 8, pp. 1799–1803, Aug. 2020.

[40] L. Huang, X. Feng, L. Zhang, L. Qian, and Y. Wu, "Multi-server multi-user multi-task computation offloading for mobile edge computing networks," *Sensors*, vol. 19, no. 6, p. 1446, 2019.

[41] M. Mukherjee, M. Guo, J. Lloret, R. Iqbal, and Q. Zhang, "Deadline-aware fair scheduling for offloaded tasks in fog computing with inter-Fog dependency," *IEEE Commun. Lett.*, vol. 24, no. 2, pp. 307–311, Jan. 2020.

[42] L. Shu, M. Mukherjee, M. Pecht, N. Crespi, and S. N. Han, "Challenges and research issues of data management in IoT for large-scale petrochemical plants," *IEEE Syst. J.*, vol. 12, no. 3, pp. 2509–2523, Sep. 2018.

[43] M. Mukherjee et al., "Latency-driven parallel task data offloading in fog computing networks for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6050–6058, Sep. 2020.

[44] M. Mukherjee, S. Kumar, M. Shojafar, Q. Zhang, and C. X. Mavromoustakis, "Joint task offloading and resource allocation for delay-sensitive fog networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.

[45] S. Singh and S. Tripathi, "A security-driven scheduling model for delay-sensitive tasks in fog networks," in *Proc. Adv. Comput. Informat. Netw. Cybersecurity*, 2022, pp. 781–807.

[46] C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo, "Computation offloading in mobile edge computing networks: A survey," *J. Netw. Comput. Appl.*, vol. 202, Jun. 2022, Art. no. 103366.

[47] M. Aazam, S. U. Islam, S. T. Lone, and A. Abbas, "Cloud of Things (CoT): Cloud-fog-IoT task offloading for sustainable Internet of Things," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 1, pp. 87–98, Jan.–Mar. 2022.

[48] U. Awada and J. Zhang, "Edge federation: A dependency-aware multi-task dispatching and co-location in federated edge container-instances," in *Proc. IEEE Int. Conf. Edge Comput. (EDGE)*, 2020, pp. 91–98.

[49] X. Cao, G. Tang, D. Guo, Y. Li, and W. Zhang, "Edge federation: Towards an integrated service provisioning model," *IEEE/ACM Trans. Netw.*, vol. 28, no. 3, pp. 1116–1129, Jun. 2020.

[50] U. Ahmed, I. Petri, and O. Rana, "Edge-cloud resource federation for sustainable cities," *Sustain. Cities Soc.*, vol. 82, Jul. 2022, Art. no. 103887.

[51] A. Kalita, M. Gurusamy, and M. Khatua, "A gaming and trust model based counter measure for DIS attack on 6TiSCH IoT networks," *IEEE Internet Things J.*, vol. 10, no. 11, pp. 9727–9737, Jun. 2022.

[52] A. Hazra, P. Choudhary, and O. Vivek, "An advance mobility management scheme in wireless network," in *Proc. 9th Int. Conf. Comput. Commun. Netw. Technol. (ICCCNT)*, 2018, pp. 1–5.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

[53] A. Dunkels, B. Gronvall, and T. Voigt, "Contiki-a lightweight and flexible operating system for tiny networked sensors," in *Proc. 29th Annu. IEEE Int. Conf. local Comput. Netw.*, 2004, pp. 455–462.

[54] P. Levis et al., "TinyOS: An operating system for sensor networks," in *Ambient Intelligence*. Berlin, Germany: Springer, 2005, pp. 115–148. [Online]. Available: https://doi.org/10.1007/3-540-27139-2_7

[55] C. Perera, P. P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context-aware dynamic discovery and configuration of 'Things' in smart environments," in *Big Data and Internet of Things: A Roadmap for Smart Environments*, N. Bessis and C. Dobre, Eds. Cham, Switzerland: Springer, 2014, pp. 215–241. [Online]. Available: https://doi.org/10.1007/978-3-319-05029-4_9

[56] E. Baccelli, O. Hahm, M. Günes, M. Wählisch, and T. C. Schmidt, "RIOT OS: Towards an OS for the Internet of Things," in *Proc. IEEE Conf. Comput. Commun. workshops (INFOCOM WKSHPS)*, 2013, pp. 79–80.

[57] E. Ferro and F. Potorti, "Bluetooth and Wi-Fi wireless protocols: A survey and a comparison," *IEEE Wireless Commun.*, vol. 12, no. 1, pp. 12–26, Feb. 2005.

[58] U. Raza, P. Kulkarni, and M. Sooriyabandara, "Low power wide area networks: An overview," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 855–873, 2nd Quart., 2017.

[59] R. Want, "An introduction to RFID technology," *IEEE Pervasive Comput.*, vol. 5, no. 1, pp. 25–33, Jan.–Mar. 2006.

[60] R. Want, "Near field communication," *IEEE Pervasive Comput.*, vol. 10, no. 3, pp. 4–7, Oct. 2011.

[61] R. S. Kshetrimayum, "An introduction to UWB communication systems," *IEEE Potentials*, vol. 28, no. 2, pp. 9–13, Jan. 2009.

[62] G. V. Crosby and F. Vafa, "Wireless sensor networks and LTE-A network convergence," in *Proc. 38th Annu. IEEE Conf. Local Comput. Netw.*, 2013, pp. 731–734.

[63] A. Ghosh, R. Ratasuk, B. Mondal, N. Mangalvedhe, and T. Thomas, "LTE-advanced: Next-generation wireless broadband technology," *IEEE Wireless Commun.*, vol. 17, no. 3, pp. 10–22, Jun. 2010.

[64] K. Pilkington, *Revolv Teams Up With Home Depot to Keep Your House Connected*, Centre Nat. d'Etudes des Telecommun., San Francisco, CA, USA, 2014.

[65] U. Rushden, *Belkin Brings Your Home to Your Fingertips With WeMo Home Automation System*. Segundo, CA, USA: Press Room Belkin, 2012.

[66] H. Wu and K. Wolter, "Stochastic analysis of delayed mobile offloading in heterogeneous networks," *IEEE Trans. Mobile Comput.*, vol. 17, no. 2, pp. 461–474, Feb. 2018.

[67] G. Gao, M. Xiao, J. Wu, K. Han, L. Huang, and Z. Zhao, "Opportunistic mobile data offloading with deadline constraints," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 12, pp. 3584–3599, Dec. 2017.

[68] Z. Hu, Z. Lu, X. Wen, and Q. Li, "Stochastic-geometry-based performance analysis of delayed mobile data offloading with mobility prediction in dense IEEE 802.11 networks," *IEEE Access*, vol. 5, pp. 23060–23068, 2017.

[69] C.-F. Liu, M. Bennis, M. Debbah, and H. V. Poor, "Dynamic task offloading and resource allocation for ultra-reliable low-latency edge computing," *IEEE Trans. Commun.*, vol. 67, no. 6, pp. 4132–4150, Jun. 2019.

[70] J. Feng, Q. Pei, F. R. Yu, X. Chu, and B. Shang, "Computation offloading and resource allocation for wireless powered mobile edge computing with latency constraint," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1320–1323, Oct. 2019.

[71] J. Wang, K. Liu, B. Li, T. Liu, R. Li, and Z. Han, "Delay-sensitive multi-period computation offloading with reliability guarantees in fog networks," *IEEE Trans. Mobile Comput.*, vol. 19, no. 9, pp. 2062–2075, Sep. 2020.

[72] M. A. Ferrag, M. Derdour, M. Mukherjee, A. Derhab, L. Maglaras, and H. Janicke, "Blockchain technologies for the Internet of Things: Research issues and challenges," *IEEE Internet Things J.*, vol. 6, no. 2, pp. 2188–2204, Apr. 2019.

[73] L. Yang, B. Liu, J. Cao, Y. Sahni, and Z. Wang, "Joint computation partitioning and resource allocation for latency sensitive applications in mobile edge clouds," *IEEE Trans. Services Comput.*, vol. 14, no. 5, pp. 1439–1452, Sep./Oct. 2021.

[74] Y. Wu, B. Shi, L. P. Qian, F. Hou, J. Cai, and X. Shen, "Energy-efficient multi-task multi-access computation offloading via NOMA transmission for IoTs," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4811–4822, Jul. 2020.

[75] H. A. Alameddine, S. Sharafeddine, S. Sebbah, S. Ayoubi, and C. Assi, "Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 3, pp. 668–682, Mar. 2019.

[76] W. Zhang, Z. Zhang, S. Zeadally, H.-C. Chao, and V. Leung, "MASM: A multiple-algorithm service model for energy-delay optimization in edge artificial intelligence," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4216–4224, Jul. 2019.

[77] S. Singh and S. Pal, "SDTS: Security driven task scheduling algorithm for real-time applications using fog computing," *IETE J. Res.*, to be published.

[78] Q. Wang, S. Guo, J. Liu, and Y. Yang, "Energy-efficient computation offloading and resource allocation for delay-sensitive mobile edge computing," *Sustain. Comput. Inf. Syst.*, vol. 21, pp. 154–164, Mar. 2019.

[79] C. Zhu, Y.-H. Chiang, A. Mehrabi, Y. Xiao, A. Ylä-Jääski, and Y. Ji, "Chameleon: Latency and resolution aware task offloading for visual-based assisted driving," *IEEE Trans. Veh. Technol.*, vol. 68, no. 9, pp. 9038–9048, Sep. 2019.

[80] P. Barnaghi, W. Wang, C. Henson, and K. Taylor, "Semantics for the Internet of Things: Early progress and back to the future," *Int. J. Semantic Web Inf. Syst.*, vol. 8, no. 1, pp. 1–21, 2012.

[81] J. Schneider, T. Kamiya, D. Peintner, and R. Kyusakov, "Efficient XML interchange (EXI) format 1.0," W3C, Cambridge, MA, USA, 2011.

[82] A. Sharma and L. K. Awasthi, "Ob-EID: Obstacle aware event information dissemination for SDN enabled vehicular network," *Comput. Netw.*, vol. 216, Oct. 2022, Art. no. 109257.

[83] Z. Xiong, S. Feng, W. Wang, D. Niyato, P. Wang, and Z. Han, "Cloud/fog computing resource management and pricing for blockchain networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4585–4600, Jun. 2019.

[84] X. Chen, Y. Cai, Q. Shi, M.-J. Zhao, B. Champage, and L. Hanzo, "Efficient resource allocation for relay-assisted computation offloading in mobile edge computing," 2019, *arXiv:1909.00478*.

[85] K. Guo, M. Sheng, T. Q. Quek, and Z. Qiu, "Task offloading and scheduling in fog RAN: A parallel communication and computation perspective," *IEEE Wireless Commun. Lett.*, vol. 9, no. 2, pp. 215–218, Feb. 2020.

[86] C. R. Chowdhury, S. Misra, C. Mandal, and S. Bera, "SeamFlow: Seamless flow forwarding in energy harvesting-enabled access points of SDWLAN," *IEEE Trans. Sustain. Comput.*, vol. 8, no. 1, pp. 94–108, Jan.–Mar. 2023.

[87] J. Zhao, Q. Li, Y. Gong, and K. Zhang, "Computation offloading and resource allocation for cloud assisted mobile edge computing in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7944–7956, Aug. 2019.

[88] J. Yan, S. Bi, Y.-J. A. Zhang, and M. Tao, "Optimal task offloading and resource allocation in mobile-edge computing with inter-user task dependency," *IEEE Trans. Wireless Commun.*, vol. 19, no. 1, pp. 235–250, Jan. 2020.

[89] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," *J. Commun. Netw.*, vol. 22, no. 1, pp. 1–11, Feb. 2020.

[90] A. Hazra, M. Adhikari, T. Amgoth, and S. N. Srirama, "Joint computation offloading and scheduling optimization of IoT applications in fog networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 3266–3278, Oct.–Dec. 2020.

[91] L. P. Qian, B. Shi, Y. Wu, B. Sun, and D. H. Tsang, "NOMA enabled mobile edge computing for Internet of Things via joint communication and computation resource allocations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 718–733, Jan. 2020.

[92] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.

[93] J. Yao and N. Ansari, "QoS-aware fog resource provisioning and mobile device power control in IoT networks," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 1, pp. 167–175, Mar. 2019.

[94] Z. Zhou, P. Liu, J. Feng, Y. Zhang, S. Mumtaz, and J. Rodriguez, "Computation resource allocation and task assignment optimization in vehicular fog computing: A contract-matching approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3113–3125, Apr. 2019.

[95] P. K. Deb, S. Misra, and A. Mukherjee, "Latency-aware horizontal computation offloading for parallel processing in fog-enabled IoT," *IEEE Syst. J.*, vol. 16, no. 2, pp. 2537–2544, Jun. 2022.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

[96] J. Kim, T. Ha, W. Yoo, and J.-M. Chung, "Task popularity-based energy minimized computation offloading for fog computing wireless networks," *IEEE Wireless Commun. Lett.*, vol. 8, no. 4, pp. 1200–1203, Aug. 2019.

[97] Y. Chen, N. Zhang, Y. Zhang, X. Chen, W. Wu, and X. S. Shen, "TOFFEE: Task offloading and frequency scaling for energy efficiency of mobile devices in mobile edge computing," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1634–1644, Oct.–Dec. 2021.

[98] D. K. Sah, S. Srivastava, R. Kumar, and T. Amgoth, "An energy efficient coverage aware algorithm in energy harvesting wireless sensor networks," *Wireless Netw.*, vol. 29, pp. 1175–1195, Jan. 2023.

[99] Q. Lin, F. Wang, and J. Xu, "Optimal task offloading scheduling for energy efficient D2D cooperative computing," *IEEE Commun. Lett.*, vol. 23, no. 10, pp. 1816–1820, Jun. 2019.

[100] X. Zhang, P. Huang, L. Guo, and Y. Fang, "Social-aware energy-efficient data offloading with strong stability," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1515–1528, Aug. 2019.

[101] F. Shan, J. Luo, J. Jin, and W. Wu, "Offloading delay constrained transparent computing tasks with energy-efficient transmission power scheduling in wireless IoT environment," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4411–4422, Jun. 2019.

[102] Z. Wei, B. Zhao, J. Su, and X. Lu, "Dynamic edge computation offloading for Internet of Things with energy harvesting: A learning method," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4436–4447, Jun. 2019.

[103] S. Sarkar, R. Wankar, S. N. Srirama, and N. K. Suryadevara, "Serverless management of sensing systems for fog computing framework," *IEEE Sensors J.*, vol. 20, no. 3, pp. 1564–1572, Feb. 2020.

[104] Y. Kim, H.-W. Lee, and S. Chong, "Mobile computation offloading for application throughput fairness and energy efficiency," *IEEE Trans. Wireless Commun.*, vol. 18, no. 1, pp. 3–19, Jan. 2019.

[105] C. You, K. Huang, H. Chae, and B.-H. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.

[106] S. Lagén, A. Pascual-Iserte, O. Munoz, and J. Vidal, "Energy efficiency in latency-constrained application offloading from mobile clients to multiple virtual machines," *IEEE Trans. Signal Process.*, vol. 66, no. 4, pp. 1065–1079, Feb. 2018.

[107] W. Cao, G. Feng, S. Qin, and M. Yan, "Cellular offloading in heterogeneous mobile networks with D2D communication assistance," *IEEE Trans. Veh. Technol.*, vol. 66, no. 5, pp. 4245–4255, May 2016.

[108] B. Chen, C. Yang, and A. F. Molisch, "Cache-enabled device-to-device communications: Offloading gain and energy cost," *IEEE Trans. Wireless Commun.*, vol. 16, no. 7, pp. 4519–4536, Jul. 2017.

[109] C. Liu, C. He, and W. Meng, "A tractable multi-RATs offloading scheme on D2D communications," *IEEE Access*, vol. 5, pp. 20841–20851, 2017.

[110] H. Guo and J. Liu, "Collaborative computation offloading for multiaccess edge computing over fiber–wireless networks," *IEEE Trans. Veh. Technol.*, vol. 67, no. 5, pp. 4514–4526, May 2018.

[111] J. Xie, Y. Jia, Z. Chen, Z. Nan, and L. Liang, "Efficient task completion for parallel offloading in vehicular fog computing," *China Commun.*, vol. 16, no. 11, pp. 42–55, 2019.

[112] I. Sarkar, M. Adhikari, S. Kumar, and V. G. Menon, "Deep reinforcement learning for intelligent service provisioning in software-Defined industrial fog networks," *IEEE Internet Things J.*, vol. 9, no. 18, pp. 16953–16961, Sep. 2022.

[113] M. Aazam, K. A. Harras, and S. Zeadally, "Fog computing for 5G tactile Industrial Internet of Things: QoE-aware resource allocation model," *IEEE Trans. Ind. Informat.*, vol. 15, no. 5, pp. 3085–3092, May 2019.

[114] M. Adhikari, T. Amgoth, and S. N. Srirama, "Multi-objective scheduling strategy for scientific workflows in cloud environment: A firefly-based approach," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106411.

[115] X. Sun, N. Ansari, and R. Wang, "Optimizing resource utilization of a data center," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 4, pp. 2822–2846, 1st Quart., 2016.

[116] S. Singh and S. Tripathi, "SLOPE: Secure and load optimized packet scheduling model in a grid environment," *J. Syst. Architect.*, vol. 91, pp. 41–52, Nov. 2018.

[117] G.-L. Huang, A. Zaslavsky, S. W. Loke, A. Abkenar, A. Medvedev, and A. Hassani, "Context-aware machine learning for intelligent transportation systems: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 1, pp. 17–36, Jan. 2023.

[118] A. Hamrouni, A. Khanfor, H. Ghazzai, and Y. Massoud, "Context-aware service discovery: Graph techniques for IoT network learning and socially connected objects," *IEEE Access*, vol. 10, pp. 107330–107345, 2022.

[119] S. Chirila, C. Lemnaru, and M. Dinsoreanu, "Semantic-based IoT device discovery and recommendation mechanism," in *Proc. IEEE 12th Int. Conf. Intell. Comput. Commun. Process. (ICCP)*, 2016, pp. 111–116.

[120] P. Joshi, A. Kalita, and M. Gurusamy, "Reliable and efficient data collection in UAV-based IoT networks." 2023. [Online]. Available: https://arxiv.org/abs/2311.05303

[121] C. Perera, P. P. Jayaraman, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Context-aware dynamic discovery and configuration of 'things' in smart environments," in *Big Data and Internet of Things: A Roadmap for Smart Environments*. Cham, Switzerland: Springer, 2014, pp. 215–241.

[122] M. Ishino, Y. Koizumi, and T. Hasegawa, "Leveraging proximity services for relay device discovery in user-provided IoT networks," in *Proc. IEEE 2nd World Forum Internet Things (WF-IoT)*, 2015, pp. 553–558.

[123] S. Singh, S. Tripathi, and N. Kumar, "An enhanced security-aware dynamic packet scheduling scheme for wireless networks using intelligent time slice-based krill herd algorithm," *J. Electromagn. Waves Appl.*, vol. 32, no. 16, pp. 2135–2156, 2018.

[124] A. Kalita, A. Brighente, M. Khatua, and M. Conti, "Effect of DIS attack on 6TiSCH network formation," *IEEE Commun. Lett.*, vol. 26, no. 5, pp. 1190–1193, 2022.

[125] L. Li, Q. Guan, L. Jin, and M. Guo, "Resource allocation and task offloading for heterogeneous real-Time tasks with uncertain duration time in a fog queueing system," *IEEE Access*, vol. 7, pp. 9912–9925, 2019.

[126] N. Waqar, S. A. Hassan, A. Mahmood, K. Dev, D.-T. Do, and M. Gidlund, "Computation offloading and resource allocation in MEC-enabled integrated aerial-terrestrial vehicular networks: A reinforcement learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 11, pp. 21478–21491, Nov. 2022.

[127] A. Hazra, P. Choudhary, and M. S. Singh, "Recent advances in deep learning techniques and its applications: An overview," in *Proc. Adv. Biomed. Eng. Technol.*, 2021, pp. 103–122.

[128] M. Adhikari and S. N. Srirama, "Multi-objective accelerated particle swarm optimization with a container-based scheduling for Internet-of-Things in cloud environment," *J. Netw. Comput. Appl.*, vol. 137, pp. 35–61, Jun. 2019.

[129] X. Shen, X. Song, X. Meng, and C. Jia, "Edge computing sever selection in fog radio access networks," in *Proc. IEEE/CIC Int. Conf. Commun. China (ICCC Workshops)*, 2018, pp. 287–291.

[130] Z. Liao et al., "Distributed probabilistic offloading in edge computing for 6G-Enabled massive Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5298–5308, Apr. 2021.

[131] J. Wang et al., "A probability preferred priori offloading mechanism in mobile edge computing," *IEEE Access*, vol. 8, pp. 39758–39767, 2020.

[132] Z. Xue, C. Liu, C. Liao, G. Han, and Z. Sheng, "Joint service caching and computation offloading scheme based on deep reinforcement learning in vehicular edge computing systems," *IEEE Trans. Veh. Technol.*, vol. 72, no. 5, pp. 6709–6722, May 2023.

[133] B. Chen, J. Wan, L. Shu, P. Li, M. Mukherjee, and B. Yin, "Smart factory of industry 4.0: Key technologies, application case, and challenges," *IEEE Access*, vol. 6, pp. 6505–6519, 2018.

[134] H. Verma, N. Chauhan, N. Chand, and L. K. Awasthi, "Buffer-loss estimation to address congestion in 6LoWPAN based resource-restricted 'Internet of Healthcare Things' network," *Comput. Commun.*, vol. 181, pp. 236–256, Jan. 2022.

[135] S. Nandy, A. Hazra, M. Adhikari, and D. Puthal, "Nanorobot-based intelligent symptoms analysis and recommendation framework in edge networks," *IEEE J. Biomed. Health Inform.*, early access, May 23, 2023, doi: 10.1109/JBHI.2023.3278991.

[136] H. K. Tripathy, S. Mishra, S. Suman, A. Nayyar, and K. S. Sahoo, "Smart COVID-shield: An IoT driven reliable and automated prototype model for COVID-19 symptoms tracking," *Computing*, vol. 104, pp. 1233–1254, Jan. 2022.

[137] K. Doulani, M. Adhikari, and A. Hazra, "Edge-based smart health monitoring device for infectious disease prediction using biosensors," *IEEE Sensors J.*, vol. 23, no. 17, pp. 20215–20222, Sep. 2023.

[138] M. Adhikari, A. Hazra, and S. Nandy, "Deep transfer learning for communicable disease detection and recommendation in edge networks," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 20, no. 4, pp. 2468–2479, Jul./Aug. 2023.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.

[139] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of Industrial Internet of Things security: Requirements and fog computing opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2489–2520, 4th Quart., 2020.
[140] R. da Rosa Righi, E. Correa, M. M. Gomes, and C. A. da Costa, "Enhancing performance of IoT applications with load prediction and cloud elasticity," *Future Gener. Comput. Syst.*, vol. 109, pp. 689–701, Aug. 2020.
[141] P. Cruz, N. Achir, and A. C. Viana, "On the edge of the deployment: A survey on multi-access edge computing," *ACM Comput. Surveys*, vol. 55, no. 5, pp. 1–34, 2022.
[142] K. Dev, S. A. Khowaja, P. K. Sharma, B. S. Chowdhry, S. Tanwar, and G. Fortino, "DDI: A novel architecture for joint active user detection and IoT device identification in grant-free NOMA systems for 6G and beyond networks," *IEEE Internet Things J.*, vol. 9, no. 4, pp. 2906–2917, Feb. 2022.
[143] Z. Yanhao, N. V. Abhishek, and M. Gurusamy, "RAVEN: Resource allocation using reinforcement learning for vehicular edge computing networks," *IEEE Commun. Lett.*, vol. 26, no. 11, pp. 2636–2640, Nov. 2022.
[144] M. Serrano, P. Barnaghi, F. Carrez, P. Cousin, O. Vermesan, and P. Friess, *Internet of Things IoT Semantic Interoperability: Research Challenges, Best Practices, Recommendations and Next Steps*, IERC, Lyon, France, 2015.
[145] M. Adhikari and A. Hazra, "6G-enabled ultra-reliable low-latency communication in edge networks," *IEEE Commun. Stand. Mag.*, vol. 6, no. 1, pp. 67–74, Mar. 2022.
[146] Z. Han, H. Tan, X.-Y. Li, S. H.-C. Jiang, Y. Li, and F. C. M. Lau, "OnDisc: Online latency-sensitive job dispatching and scheduling in heterogeneous edge-clouds," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2472–2485, Dec. 2019.
[147] D. He, S. Zeadally, B. Xu, and X. Huang, "An efficient identity-based conditional privacy-preserving authentication scheme for vehicular ad hoc networks," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2681–2691, Dec. 2015.

Abhishek Hazra (Member, IEEE) received the B.Tech. degree from the National Institute of Technology Agartala, Agartala, India, in 2014, the M.Tech. degree in computer science and engineering from the National Institute of Technology Manipur, Imphal, India, in 2018, and the Ph.D. degree from the Indian Institute of Technology (Indian School of Mines) Dhanbad, Dhanbad, India, in 2022.

He is a Postdoctoral Research Fellow with the Communications and Networks Lab, Department of Electrical and Computer Engineering, National University of Singapore, Singapore. His research area of interest includes IoT, fog/edge computing, distributed learning, and Industry 4.0.

Dr. Hazra currently serves as an Area Editor for *Computer Communication* and *Physical Communication*.

Alakesh Kalita (Member, IEEE) received the Ph.D. degree in computer science and engineering from the Indian Institute of Technology Guwahati, Guwahati, India, in 2022.

He is currently a University Lecturer with Singapore University of Technology and Design, Singapore. Prior to this, he was a Postdoctoral Research Fellow with the National University of Singapore, Singapore. His research interests include Internet of Things and edge/cloud computing.

Dr. Kalita received the prestigious Indian National INAE's Best Project Award 2022 for his Ph.D. thesis.

Mohan Gurusamy (Senior Member, IEEE) received the Ph.D. degreedegree in computer science and engineering from the Indian Institute of Technology Madras, Chennai, India, in 2000.

He joined the National University of Singapore, Singapore, where he is currently an Associate Professor with the Department of Electrical and Computer Engineering. He has about 220 publications to his credit, including two books and three book chapters in the area of optical networks. His research experience and interests are in the areas of IoT, 5G networks, software-defined networks, cloud computing, and optical networks.

Dr. Gurusamy served as the TPC Co-Chair for IEEE GLOBECOM 2019 (ONS) and IEEE ICC 2008 (ONS). He has served as an Editor for IEEE TRANSACTIONS ON CLOUD COMPUTING and is serving on the editorial board for *Computer Networks* and *Photonic Network Communications*.

Authorized licensed use limited to: Singapore University of Technology & Design. Downloaded on April 29,2024 at 06:50:37 UTC from IEEE Xplore. Restrictions apply.