IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 3 May 2025, accepted 13 May 2025, date of publication 19 May 2025, date of current version 27 May 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3571394

RESEARCH ARTICLE

Dynamic Threshold-Based Resource Management for Fog Computing Environments

JUI-PIN YANG, (Member, IEEE)
College of Marine Resources and Engineering, National Penghu University of Science and Technology, Penghu, Magong 880011, Taiwan
e-mail: juipinyang@gmail.com

ABSTRACT Fog computing extends cloud services to the network edge, thereby reducing latency and bandwidth usage for time-sensitive applications. However, the limited computational capacity, memory, and bandwidth of fog nodes present significant challenges for efficient resource management. This paper proposes a Dynamic Threshold-based Resource Management (DTRM) strategy that dynamically adjusts resource allocation thresholds to prioritize real-time (RT) requests while minimizing the redirection of non-real-time (NRT) requests to cloud servers. To further support balanced workload distribution, a Quota-Based Round-Robin (QRR) scheduler is introduced, ensuring fairness and low computational overhead across fog nodes. Extensive experimental evaluations demonstrate that DTRM significantly reduces the resource redirection count compared to Static Threshold (ST), Deficit Round Robin (DRR), and Best-Effort (BE) schemes. Moreover, DTRM improves load balancing and resource usage, offering a scalable and adaptive solution for dynamic and bursty request patterns in fog computing environments. These results highlight the potential of DTRM to enhance the overall performance, responsiveness, and efficiency of fog computing, particularly in the context of modern IoT applications.

INDEX TERMS Dynamic threshold, resource management, fog computing, quota-based round robin, load balancing.

I. INTRODUCTION

The rapid growth of Internet of Things (IoT) applications generates vast amounts of data, posing significant challenges to traditional cloud computing environments, including network bandwidth limitations, latency issues, and security concerns [1]. Software-as-a-Service (SaaS) is one of the primary applications of cloud computing; however, as user interactions increase, the complexity of cloud-based processes also escalates. Consequently, providing efficient services that support Customer Relationship Management (CRM) applications remains a challenge in cloud environments [2]. A Virtual Machine (VM) allocation algorithm based on the best-fit decreasing approach has been proposed to reduce energy consumption and Service Level Agreement Violation (SLAV), thereby improving VM allocation efficiency [3].

Fog computing integrates processing, networking, and storage elements to bring computational services closer to IoT devices. As a result, it reduces energy consumption, bandwidth usage, and latency compared to traditional cloud computing [4]. However, fog nodes have limited resources, necessitating more effective resource management strategies within fog computing environments. The centralized nature of cloud architecture introduces heavy data transmission loads and increased latency, making it impractical for many real-time applications. This challenge underscores the need for fog computing, which leverages edge nodes within the network. However, to implement fog computing effectively, managing edge nodes efficiently is crucial, making resource optimization a critical challenge [5].

Fog computing has also been increasingly adopted in healthcare applications to enhance data management and reliability. A systematic literature review on fog computing in the healthcare sector has identified key challenges related to resource management in medical applications [6]. As the adoption of IoT continues to grow, an increasing number of devices with limited hardware capabilities are becoming interconnected. Integrating these resource-constrained

The associate editor coordinating the review of this manuscript and approving it for publication was Kashif Sharif.

devices with fog computing is crucial for the future development of IoT. A classification framework based on resource requirements has been proposed to address this challenge [7]. In such an integration, services are preferentially allocated to fog nodes, with the cloud serving as a fallback when fog nodes lack the capacity to process requests.

Several architectures, infrastructures, and algorithms have been developed to optimize resource management in fog computing, as it is critical for enhancing overall system performance [8]. In distributed architectures, the increasing number of IoT devices underscores the necessity of efficient methods for allocating services to fog nodes. Processing all data in the cloud can result in significant drawbacks, including excessive bandwidth consumption, increased latency, and high energy costs. To address these issues, a hierarchical architecture has been proposed to manage resources across edge, fog, and cloud tiers [9]. Additionally, two-way communication within this architecture enables real-time decision-making. Given the vast number of IoT services, effectively distributing tasks among fog nodes is essential. Linked-microservices have been introduced to decompose and distribute services across multiple fog nodes while maintaining strong interconnectivity with partner nodes, ultimately reducing data transmission overhead [10].

Optimizing service allocation in fog computing while maintaining high resource utilization is crucial. Effectively understanding and exploiting the dynamic behavior of fog nodes presents a significant challenge. An online optimization framework has been proposed to iteratively adjust the target competitive ratio, thereby minimizing maximum communication and computation latency through optimal task distribution [11]. Furthermore, a smart online solution based on reinforcement learning has been explored to minimize makespan and average waiting time under resource constraints. To enhance efficiency, parallel multi-agent technologies have been incorporated. However, most of these solutions are primarily designed for cloud computing rather than fog computing [12]. To accommodate mobile users with diverse quality-of-service (QoS) requirements, container migration algorithms and architectures have been developed, aiming to optimize delay, power consumption, and migration costs [13].

The objective of this study is to evaluate the performance of the proposed scheme through simulation. We introduce an efficient dynamic threshold strategy that dynamically allocates optimal resources for real-time (RT) requests, thereby maintaining high quality of service (QoS). For non-real-time (NRT) requests, this strategy also minimizes the number of requests redirected to the cloud. Furthermore, a quota-based round-robin (QRR) strategy is employed to optimally distribute workloads across fog nodes, ensuring superior load balancing. The remainder of this paper is structured as follows: Section II reviews related work on resource management in fog computing. Section III introduces the proposed Dynamic Threshold-Based Resource Management (DTRM) scheme. In Section IV, we present simulation results to evaluate the performance of our approach and compare it with existing schemes, including Static Threshold (ST) and Best Effort (BE). Finally, Section V concludes the paper and discusses future research directions.

II. LITERATURE REVIEW

Task offloading is a critical aspect of resource management, making it highly relevant to modern IoT and cloud computing applications. This process can take place on IoT nodes, fog nodes, and other computational units. The decision to offload tasks is influenced by various factors, such as computational demands, load balancing, and latency reduction. Several offloading criteria related to latency, energy consumption, and load balancing have been studied [14]. However, these studies primarily focus on task offloading methods without addressing resource management techniques such as resource allocation, load balancing, and scheduling. In the context of executing services in combined edge and cloud computing environments, ensuring resource continuity is crucial for enhancing performance and resource efficiency. A layered architecture has been devised to facilitate efficient resource provisioning and selection [15]. However, this study primarily concentrates on architectural aspects, neglecting load balancing, task offloading, and application placement in terms of algorithmic resource management.

To enhance performance through efficient resource utilization, a taxonomy of resource management methods has been proposed, categorizing methods based on resource type, management goals, resource usage, and location [16]. A survey summarizes research in resource allocation, scheduling, and fault tolerance for fog computing [17]; however, it does not cover aspects such as load balancing and quality of service (QoS). Cost minimization encompasses various aspects, including resource usage, QoS, and associated revenue. The edge user allocation problem has been modeled as a bin-packing problem [18]. Furthermore, a novel approach based on the lexicographic goal programming technique has been proposed to maximize user allocation numbers while considering edge server usage and QoS maintenance. Additionally, data communication over the network has been considered. A study examines the advantages and disadvantages of load balancing mechanisms in fog networks [19], addressing critical open challenges and exploring future trends related to these mechanisms. Several studies have partially investigated the dynamic behavior of fog nodes in fog computing platforms.

In [20], a low-latency and massive-connectivity vehicular fog computing framework is proposed, aiming to minimize processing delays through centralized optimization of task allocation between user equipment and vehicles with underutilized computational resources. However, this framework is specifically tailored to vehicular environments. A mobile task offloading framework based on device-to-device (D2D) collaboration was proposed to optimize delay and energy consumption, enabling mobile users to share resources among devices [21]; however, a complete solution was not provided.

A low-complexity metaheuristic and a greedy heuristic were employed to address the joint problem of container placement and task provisioning in dynamic fog computing environments [22]. These solutions optimize the number of served end-users within a predefined delay threshold but are not applicable to practical and dynamic environments where location and behavioral patterns are unavailable.

For edge-cloud resource allocation, a novel online algorithm has been proposed to optimally solve a series of subproblems, producing feasible solutions [23]. This algorithm provides near-optimal results with a parameterized competitive ratio without requiring prior knowledge of resource prices or user mobility. In [24], an online resource allocation scheme is presented that determines the state of servers in fog nodes across different zones using a Stackelberg game. Additionally, a multi-round dynamic scheme is derived from an initial static scheme.

The authors investigated the trade-off between service latency and quality loss in task allocation to fog nodes [25]. Consequently, they proposed an event-triggered dynamic task allocation framework that optimizes performance using linear programming and binary particle swarm optimization. Computing resources are generally heterogeneous, constrained, and dynamic. Allocating resources to perform tasks while meeting QoS requirements is critical [26]. This study surveys the current state of research and identifies future directions for efficient resource management in fog computing. An efficient resource provisioning approach utilizing Bayesian learning techniques for real-time decision-making in fog resources has been proposed [27]. Additionally, an autonomous resource provisioning framework has been introduced. In summary, this study aims to improve cost efficiency, reduce latency, and enhance resource utilization. In [28], an optimal and heuristic method is proposed to minimize the cost of offloading applications. This study considers both cost and delay, contingent on selecting an appropriate set of fog nodes while deploying a virtual machine (VM) for each application.

To minimize the end-to-end latency of real-time applications, a proposed scheme comprises a greedy delay-minimizing application module selection stage and a greedy delay-minimizing application module placement stage [29]. The first stage determines the order in which application modules are placed across fog devices to maintain QoS, while the second stage selects the fog node that meets the processing, storage, and bandwidth requirements of the applications. In [30], various problems are addressed, including application placement, application offloading, and task scheduling in a fog-cloud environment, with the aim of allocating services of various IoT applications to adequate resources. Moreover, a three-tier architecture is proposed to support various IoT applications and dynamically meet QoS requirements. Additionally, various QoS parameters are mathematically formulated to achieve optimal and efficient resource provisioning.

With the growth of IoT, more devices are connected, generating vast amounts of data. Offloading computing-intensive tasks to edge devices and using cloud-based storage has become mainstream [31]. However, sending all data to the cloud raises data privacy concerns. This paper proposes a fog computing-based data synchronization architecture that offloads computing and storage tasks to fog servers to ensure data privacy. To reduce communication costs and latency, a differential synchronization algorithm is introduced. Additionally, Reed-Solomon coding is used for enhanced security. Load balancing in fog environments is crucial for resource efficiency, preventing bottlenecks, overload, and underutilization [32]. This paper proposes a dynamic resource allocation method, DRAM, to address load balancing challenges in fog environments. A system framework was proposed to analyze load balance across different computing nodes. DRAM integrates static resource allocation and dynamic service migration to achieve load balance. However, the DRAM is evaluated through simulations rather than real-world deployment, limiting its practical validation. Additionally, the study lacks a thorough analysis of network latency, communication overhead, and the impact of service migration. While it improves resource utilization and load balancing, its scalability in large-scale fog environments remains uncertain. Furthermore, the comparison is primarily against traditional methods, without considering more recent AI-driven approaches that may offer better efficiency.

Cloud computing enables the sharing of computing resources through virtualization, allowing dynamic resource allocation to meet client needs. Elastic resource allocation optimizes resource use, improving performance by minimizing response time and maximizing throughput [33]. However, the study does not comprehensively address scalability, energy efficiency, or security concerns, which are crucial for practical deployment. A more thorough comparison with state-of-the-art techniques and an empirical performance analysis would strengthen its contributions.

While various solutions exist for cloud computing, efficient solutions for fog computing are still needed. Fog computing serves as a virtualized intermediate layer between clients and the cloud, offering data, computation, storage, and networking services. This paper presents an efficient architecture and algorithm for resource provisioning in fog computing using virtualization techniques. Efficient resource provision is essential to meet user needs, guided by Service Level Agreements (SLA) and parallel processing [34]. It is unclear how well the approach performs under high user and resource loads. The impact of energy efficiency is also not considered, which is crucial for optimizing cloud infrastructures. Furthermore, the paper provides limited comparisons with other advanced scheduling methods, leaving its competitive advantages uncertain.

Based on the reviewed literature on resource management, limited research has addressed performance issues related to QoS provisioning, particularly regarding the number of real-time and non-real-time requests redirected to cloud

servers (i.e., the resource redirection count). The resource redirection count has a significant impact on QoS metrics, including latency, communication overheads, cost, and throughput. To address resource shortages in fog computing, cloud servers must provision additional resources, including enhanced computational power, storage capacity, and network bandwidth. Consequently, an efficient resource management scheme is crucial to maintaining the quality of service (QoS) in fog computing systems. Moreover, the proposed approach concurrently addresses the challenge of load balancing. Therefore, it offers a practical and effective solution for fog computing environments.

III. DYNAMIC THRESHOLD-BASED RESOURCE MANAGEMENT

Figure 1 illustrates the system architecture, in which clients submit resource allocation requests to the fog controller. The controller then assigns appropriate fog nodes to fulfill these requests based on their specific requirements. To ensure optimal quality of service (QoS), real-time (RT) requests should be allocated resources on fog nodes, as they require performance metrics such as latency, cost, and bandwidth. Therefore, RT requests should be processed by fog computing first. In contrast, non-real-time (NRT) requests are best allocated to the cloud server, as they do not have strict real-time requirements. The fog controller manages all incoming requests and determines the most efficient resource allocation for both RT and NRT requests.

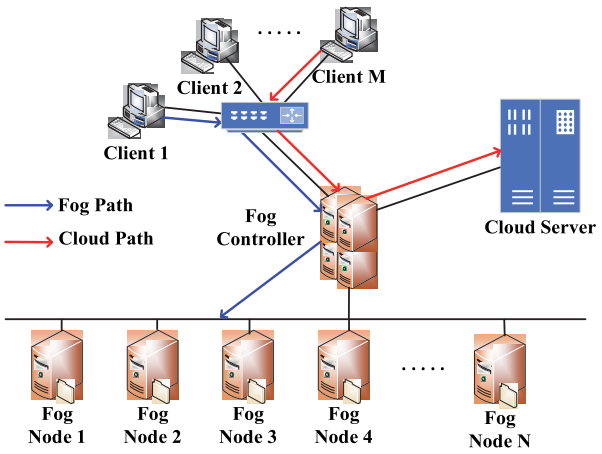


FIGURE 1. System architecture.

Figure 2 illustrates the three primary components of the Dynamic Threshold-based Resource Management (DTRM) within the fog controller: the resource estimator, request handling and quota-based round robin scheduler. The resource estimator evaluates the resources required for RT requests. Subsequently, the request handling component uses a dynamic threshold to decide whether incoming RT or NRT requests should be enqueued in the FIFO queue for advanced processing or redirected to the cloud server. Accepted requests are then processed using a QRR algorithm, which determines their scheduling. RT requests pertain to resource demands that necessitate minimal latency in processing and delivery. These requests must be addressed promptly to maintain system efficiency and fulfill user expectations, underscoring the critical importance of handling RT requests effectively to sustain QoS performance. In contrast, NRT requests do not require immediate processing and can be scheduled with greater flexibility. Therefore, it is crucial to optimize the handling of RT requests while simultaneously maximizing the number of NRT requests processed by fog nodes. Next, we sequentially introduce the three core components of the proposed strategy, starting with a detailed explanation of the resource estimator.

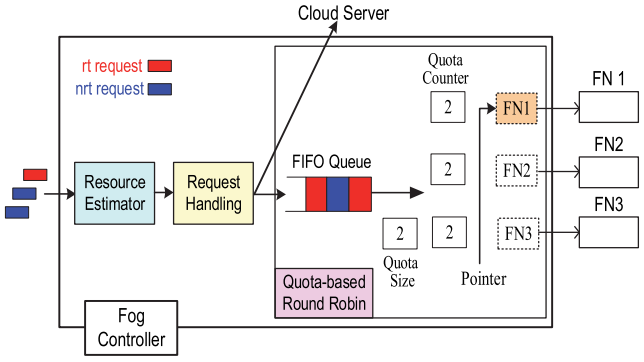


FIGURE 2. Quota-based Round Robin: At the start, all the Quota Counter variables are initialized to the resource level at the head of queue. The pointer points to the top of the fog node 1. When the first resource level is serviced, the remainder after servicing is left in the Quota Counter variable.

Let C_i denote the CPU resource (MIPS) of fog node i, M_i denote the memory resource (MB) of fog node i and B_i denote the network bandwidth resource (MB) of fog node i. Additionally, let C_b, M_b and B_b represent a basic resource allocation unit for CPU, memory and bandwidth, respectively. Therefore, the maximum number of resource unit for fog node i, denoted as R_i, can be determined using (1). Equation (1) calculates the minimum value among the three given resources, with the min function returning the smallest value.

$$R_i = \min({}^{C_i}/_{C_b}, {}^{M_i}/_{M_b}, {}^{B_i}/_{B_b}) \tag{1}$$

Next, we use (2) to estimate the number of RT requests at the beginning of the kth time interval, denoted as A_k. Similarly, A_{k-1} denotes the number of RT requests at the beginning of the (k−1)th time interval, and $A_k^{current}$ represents the current number of arriving RT requests between the (k−1)th and kth time intervals. Finally, α_k is a parameter used to balance the estimate of A_k to account for both short-term and long-term variations in the estimated value of RT requests at the beginning of the kth time interval. This ensures that the estimation remains sensitive to recent changes while also reflecting broader trends. T_d denotes the duration of each time interval.

$$A_k = \alpha_k * A_k^{current} + (1 - \alpha_k) * A_{k-1} \tag{2}$$

To achieve an accurate estimation of A_k, we adjust α_k based on the standard deviation of a moving average. Initially, we calculate the simple moving average (SMA) of the number of RT requests at the beginning of the kth time interval, denoted as SMA_k, using (3). Here, w_{size} represents the window size of SMA, indicating the number of time intervals considered in the computation of the average.

$$SMA_k = \sum_{i=k-w_{size}}^{k-1} A_i / w_{size} \tag{3}$$

Next, we calculate the standard deviation at the beginning of the kth time interval using (4), denoted as σ_k. The standard deviation of the moving average is used to improve the accuracy and consistency of the estimated values of RT requests and to manage resource allocation more effectively.

$$\sigma_k = \sqrt{\frac{1}{w_{size}} * \sum_{i=k-w_{size}}^{k-1} (A_i - SMA_i)^2} \tag{4}$$

To refine the estimated value of α_{k+1}, we introduce the adjustment factor γ using (5). Apparently, the new estimate α_{k+1} is influenced by σ_k.α_{min} is the minimum bound of α_{k+1} and α_{max} is the maximum bound of α_{k+1}; both are used to alleviate the burstiness in the number of arriving requests that cause larger variations.

$$\alpha_{k+1} = \begin{cases} \max(\alpha_{min}, \alpha_k - \gamma * \sigma_k) & A_k^{current} > A_k \\ \alpha_k & A_k^{current} = A_k \\ \min(\alpha_{max}, \alpha_k + \gamma * \sigma_k) & A_k^{current} < A_k \end{cases} \tag{5}$$

Next, we estimate the total available resources of all fog nodes at the beginning of the kth time interval, denoted as V_k, using (6). $u_{i,k}$ denotes the occupied number of resource unit for fog node i at the beginning of the kth time interval. N represents the number of fog nodes in a fog network.

$$V_k = \sum_{i=1}^{N} (R_i - u_{i,k}) \tag{6}$$

Equation (7) is used to calculate the $u_{i,k}$ where $ntc_{i,k}$ denotes the allocated number of NRT request at kth time interval for fog node i, and $tc_{i,k}$ denotes the required number of resource units for RT request at kth time interval for fog node i. Additionally, q_{nrt} denotes the required number of resource units for a NRT request, and q_{rt} denotes the required number of resource unit for a RT request.

$$u_{i,k} = ntc_{i,k} * q_{nrt} + tc_{i,k} * q_{rt} \tag{7}$$

Next, we introduce the request handler. We use (8) to calculate S_k, where S_k denotes the threshold of reserved resources for RT requests, which is the minimum value of V_k and $\beta_k * A_k * q_{rt}$. β_k is a parameter used to control the number of reserved resources based on their sufficiency. When the available resource is less than or equal to S_k, the arriving RT request will be handled in the fog controller; otherwise, NRT request will be redirected to the cloud server for advanced processing. If all resource is occupied, whether RT or NRT requests, they will both be redirected to the cloud server.

$$S_k = \min(V_k, \beta_k * A_k * q_{rt}) \tag{8}$$

When $\beta_k * A_k * q_{rt} > V_k$, it indicates that the resources are insufficient. Therefore, k_p increases by 1, with its maximum value being k_{max}. Otherwise, k_p decreases by 1, with its minimum value being 0. Here, k_p is a parameter used to record the number of occurrences when $\beta_k * A_k * q_{rt} > V_k$. Equation (9) is utilized to update k_p, which in turn adjusts β_{k+1} to achieve accurate and adequate resource allocation.

$$k_p = \begin{cases} \min(k_{max}, k_p + 1) & \beta_k * A_k * q_{rt} > V_k \\ \max(0, k_p - 1) & \beta_k * A_k * q_{rt} < V_k \end{cases} \tag{9}$$

Finally, we update β_{k+1} using (10) where w_a is a parameter utilized to adjust β_{k+1}.

$$\beta_{k+1} = (1 + w_a)^{k_p} \tag{10}$$

After processing the requests, a Quota-Based Round-Robin (QRR) scheduler is applied to achieve effective load balancing across fog nodes. Figures 2 and 3 illustrate the operations of the QRR. In Fig. 2, it is assumed that an RT request requires a quota of 2 resource units, while an NRT request requires a quota of 1 resource unit. In the FIFO queue, the RT request is at the head of the queue, and the pointer of the QRR begins at fog node 1. The available quota size of fog node 1 is sufficient to handle the RT request, making it the candidate to allocate resources for the RT request. In Fig. 3, a newly arriving RT request is enqueued in the FIFO queue, while the required resources for the previous RT request have already been allocated on fog node 1. As a result, the quota counter of fog node 1 becomes 0, prompting the pointer of the QRR to move to fog node 2. The quota size is added to the quota counter, either 1 or 2 resource units depending on the request at the head of the FIFO queue, when all quota counters are insufficient to handle the request at the head of the queue.

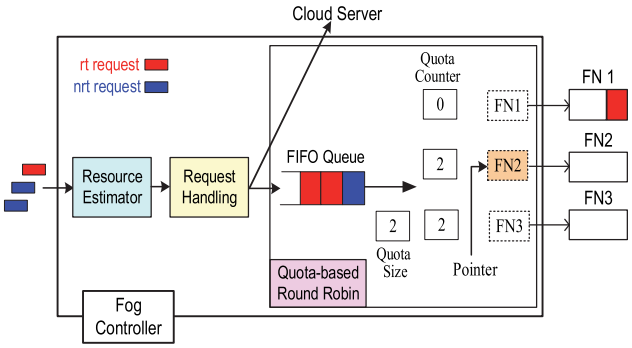


FIGURE 3. Quota-based Round Robin (2): After allocating the required resource level on fog node 1, the quota counter is 0. It could not use it the current round, since the next resource level is 1. Therefore, the pointer points to fog node 2.

The QRR scheduler maintains certain state information; however, none of it pertains to individual fog nodes. Thus, when referring to QRR as "stateless," it specifically denotes the absence of per-fog-node state tracking. In fog computing environments, real-time updates of fog node status pose several critical challenges. Frequent updates can result

in significant communication overhead, thereby increasing network traffic and latency. Moreover, the continuous exchange of data between fog nodes and the fog controller consumes valuable network resources and processing time, which can degrade overall system performance and responsiveness. Additionally, maintaining up-to-date information across all fog nodes necessitates considerable computational resources, straining processing power and memory. This continuous monitoring and updating process can reduce efficiency and pose scalability challenges as the number of fog nodes increases. Furthermore, the need for real-time updates results in increased energy consumption. Therefore, maintaining a stateless scheduling approach proves advantageous. The QRR is an advanced variant of the Deficit Round Robin (DRR) algorithm [35]. DRR is primarily used in computer networking to manage packet transmission queues. It improves upon the traditional Round Robin scheduling algorithm by effectively handling varying packet sizes while ensuring fairness and efficiency. The pseudocode for the QRR algorithm is presented in Fig. 4. The input consists of incoming requests stored in a FIFO queue, and the output is the corresponding fog node assigned to handle each request. Although we discuss RT and NRT requests as two request types, the DTRM can be easily extended to support multiple request types, each corresponding to different resource requirements. This extension enables the DTRM to accommodate varying granularity in resource demands, thereby enhancing resource utilization.

DTRM is an efficient scheduling algorithm that ensures fair distribution of resources across multiple fog nodes. It has a time complexity of O(1), meaning that each scheduling decision takes constant time, as it only involves updating and checking the quota size for each fog node. The space complexity is O(n), where n is the number of fog nodes, since each fog node needs to store its quota size and other relevant information. This makes DTRM both time-efficient and scalable for handling multiple fog nodes in fog computing.

```
Enqueue(), Dequeue() are standard queue operations.
Allocate(i) allocates resources on fog node i.
Quota(head) obtains the required quota at the head of
queue.
QuotaSize is the quantum allocate to each fog node.
QuotaCounter_i is the quota counter of fog node i.
N is the total number of fog nodes.
Initialization:
QuotaSize = Quota(head);
for(i = 0; i < N ; i++)
QuotaCounter_i = QuotaSize;
Enqueuing module: on acceptance of request r
j = ExtractType(r) //extracts the request type of request r
Enqueue(j); //enqueues request type j to queue
Dequeuing module:
while (QuotaCounter_i >0)
{
   if (QuotaCounter_i >= Quota(head))
   {
   Allocate(i);
   Dequeue(queue);
   QuotaCounter_i = QuotaCounter_i - Quota(head);
   }
   else
   break;
}
Quoting module:
k=0;
for(i = 0; i < N ; i++)
   if(QuotaCounter_i < Quota(head))
   k++;
if(k == N)
   for(i = 0; i < N ; i++)
   QuotaCounter_i = QuotaCounter_i + Quota(head);
```

FIGURE 4. Pseudocode for Quota-based Round Robin.

IV. SIMULATION RESULTS

All simulations were conducted based on the system architecture depicted in Fig. 1. The service request model, which simulates client requests, is implemented using ON-OFF models with varying parameter settings. Additionally, the simulations were performed using MATLAB. To evaluate the performance of the proposed Dynamic Threshold-based Resource Management (DTRM) approach, we analyzed the resource redirection count and resource utilization in comparison with Static Threshold (ST), Deficit Round Robin (DRR), and Best Effort (BE) under the specified system architecture. The ST approach employs a static threshold, denoted as ST_{ratio}, for each fog node i to determine the acceptance of incoming RT and NRT requests. When available resources are less than or equal to $ST_{ratio} * R_i$, only RT requests are accepted while NRT requests are redirected to the cloud server. Conversely, when resources exceed $ST_{ratio} * R_i$, both RT and NRT requests are accepted. DRR is a scheduling algorithm designed to fairly distribute network bandwidth among multiple data flows. It improves upon traditional round-robin methods by introducing a ''deficit counter'' that keeps track of leftover credits from previous rounds. This allows DRR to handle variable-sized packets efficiently and ensures fair service without complex computations. DRR is simple to implement and performs well in high-speed networks. In cases where no resources are available, both NRT and RT requests are redirected to the cloud server. The BE approach, on the other hand, imposes no constraints on RT or NRT requests; all requests are accepted until resources are fully occupied, after which both types of requests are redirected to the cloud server. Both ST and BE utilize an optimal load scheduling strategy, where the fog node with the lowest resource usage is selected to allocate resources for accepted requests. This optimal load scheduling requires significant communication overheads and frequent estimations to determine the appropriate fog node, aiming to optimize load balancing. However, this approach is impractical for implementation in the fog controller due to its high communication overheads and computational demands.

We consider ON-OFF models to represent the generation of non-real-time (NRT) and real-time (RT) requests. When a request is generated, a request ratio, denoted as θ, is used to probabilistically determine whether it is an RT request; otherwise, it is classified as an NRT request with probability (1-θ). All requests are managed by the fog controller. In the context of a fog network, the state transition probabilities and parameters are crucial in modeling request behavior within an ON-OFF model. Specifically, $P_{on-off}(i)$ represents the probability that client i transitions from the ON state to the OFF state, where no request is generated.

Conversely, $P_{off-on}(i)$ denotes the probability that client i transitions from the OFF state to the ON state. These probabilities are further generalized as the ON-OFF transition parameter on_off_pb and the OFF-ON transition parameter off_on_pb. For instance, $P_{on-off}(i) = 1/\text{on\_off\_pb}$ and $P_{off-on}(i) = 1/\text{off\_on\_pb}$. These probabilities and parameters are essential in accurately modeling the dynamics of requests within a fog network. By understanding these transitions, we can better predict and manage the behavior of NRT and RT requests, which aids in the analysis of the performance of our proposed approach. Additionally, it helps in identifying potential bottlenecks. Effective utilization of these state transition probabilities allows for more precise simulation and evaluation, ultimately leading to enhanced performance and scalability of the fog network.

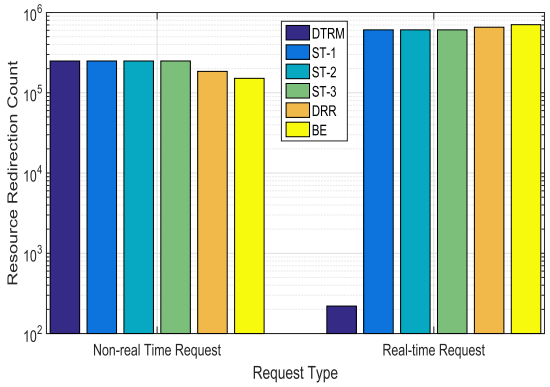
Unless otherwise specified, we used the following parameter settings in all experiments. The parameters for the DTRM are set as follows: $\alpha_0 = 0.60$, $\alpha_{min} = 0.1$, $\alpha_{max} = 0.9$, $\beta_0 = 1.0$, $\theta = 0.75$, $\gamma = 0.02$, $w_{size} = 3$, $w_a = 0.05$, $k_{max} = 5$, $C_b = 1000$ (MIPS), $M_b = 500$ (MB), $B_b = 500$ (MB) and $T_d = 1$ minute. The total simulation duration is 144 hours. In addition, each fog node has the same resources as fog node i, that is, $C_i = 30000$ (MIPS), $M_i = 15000$ (MB) and $B_i = 15000$ (MB) and then $R_i = 30$. In the DRR scheduling mechanism, each fog node is assigned a quantum size of 2 resource units. In the ST approach, ST-1 means that $ST_{ratio} = 0.3$, ST-2 means that $ST_{ratio} = 0.6$ and finally ST-3 means that $ST_{ratio} = 0.9$. When $ST_{ratio} = 0.3$, it indicates that 30% of the resources are reserved for RT requests. In other words, when resource usage exceeds 70%, the NRT requests will be handled by the cloud server instead of the fog nodes. There are 128 clients generating requests and 16 fog nodes. Finally, the transition parameter settings of the ON-OFF models are as follows: on_off_pb = 180, off_on_pb = 20. In each time interval, the allocated resources for NRT or RT requests will be released proportionally to the ratio of residing NRT and RT requests.

In the first experiment, we considered a scenario with 128 clients and 16 fog nodes. The experimental results are illustrated in Fig. 5(a) and Fig. 5(b). In Fig. 5(a), the DTRM demonstrates the lowest resource redirection count compared to ST-1, ST-2, ST-3, DRR and BE. This is attributed to the DTRM's ability to accurately estimate the required resources for RT requests, thereby reserving sufficient and precise resources. ST-1, ST-2, and ST-3 exhibit similar resource redirection counts due to the heavy and varying load of RT requests, rendering the values of ST_{ratio} ineffective. While the deficit counter mechanism in DRR facilitates fairer bandwidth allocation, it is less effective than DTRM in minimizing resource redirection count. Consequently, the number of resource redirection count under DRR approaches the static threshold. The BE exhibits the highest resource redirection count for RT requests, as it does not reserve resources specifically for them. Conversely, BE shows the lowest redirection count for NRT requests. These results clearly demonstrate that DTRM effectively prioritizes RT requests over NRT requests, as reflected by the significantly lower redirection count for RT request. In addition, DRR, ST-1, ST-2, ST-3, and BE exhibit higher resource redirection counts for RT requests compared to NRT requests. This discrepancy is primarily attributed to the high proportion of RT requests, which constitute 75% of the total generated requests. Furthermore, frequent resource saturation results in more RT requests being redirected to cloud servers due to insufficient fog resources.

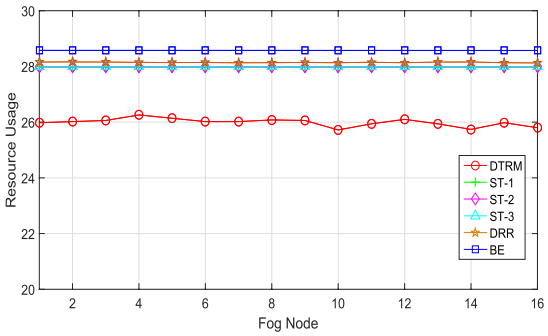
In Fig. 5(b), we evaluated the performance of resource usage. The QRR mechanism in DTRM demonstrates the lowest resource usage among all compared schemes, including DRR, ST-1, ST-2, ST-3, and BE. Given that DTRM exhibits lower resource usage than both the ST variants and BE, it is reasonable to infer that it also contributes to reduced energy consumption. The BE exhibits the highest resource usage due to the absence of constraints on incoming request acceptance. In contrast, the ST variants demonstrate only marginal improvements in resource consumption, primarily owing to inadequate regulation of NRT requests. The DRR performs similarly to the ST variants, as it also lacks effective control over NRT requests. Maintaining low resource usage is essential for ensuring rapid request processing and response time, which in turn reduces latency and enhances the overall performance of fog computing. Given the geographically distributed nature of fog nodes, efficient resource management at each fog node not only improves system performance but also minimizes operational costs and enhances scalability—critical factors for supporting modern Internet of Things (IoT) applications.

In the second experiment, we considered a different scenario involving 256 clients and 16 fog nodes. Additionally, we modified the parameter settings as follows: $\theta = 0.60$, on_off_pb = 50, off_on_pb = 750 to generate different request patterns. The experimental results are illustrated in Fig. 6(a) and Fig. 6(b). In Fig. 6(a), the resource redirection count of the DTRM is higher than that in Fig. 5(a) because the number of clients increases from 128 to 256, resulting in greater request burstiness. Consequently, this slightly reduces the estimation accuracy of resource estimator. ST-3 has a lower resource redirection count compared to ST-1 and ST-2 because more resources are reserved for RT requests. However, this also causes ST-3 to have a higher resource redirection count for NRT requests than ST-1 and ST-2. The resource redirection count for RT requests in the DRR is higher than that observed in the ST variants. However, for NRT requests, DRR exhibits a lower redirection count compared to the ST variants. Overall, the performance of DRR surpasses only that of the BE, indicating limited improvement relative to more controlled schemes. The DTRM exhibits the lowest resource redirection count for RT requests among all evaluated approaches, indicating its superior efficiency in handling time-sensitive services.

In Fig. 6(b), we evaluated the performance of resource usage. The resource usage in all schemes is lower than in the previous scenario due to the reduced intensity of request generation. ST-3 exhibits the lowest resource usage because

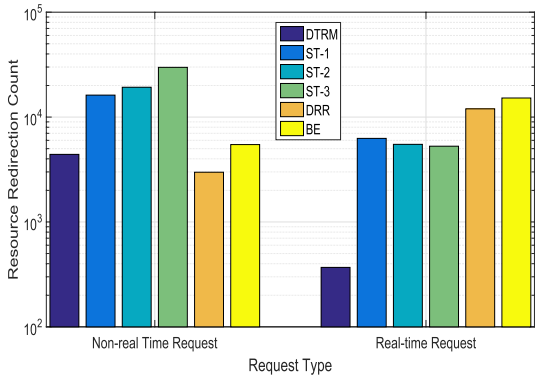


(a) Resource redirection count

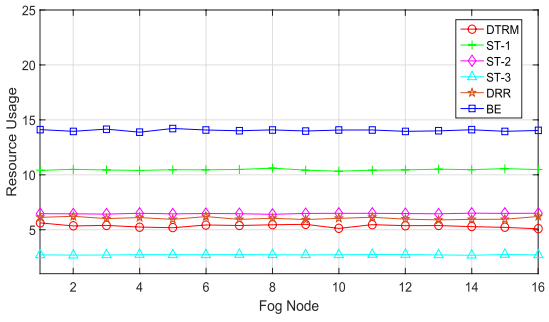


(b) Resource usage

FIGURE 5. Performance comparisons of different schemes with 128 clients and 16 fog nodes.



(a) Resource redirection count



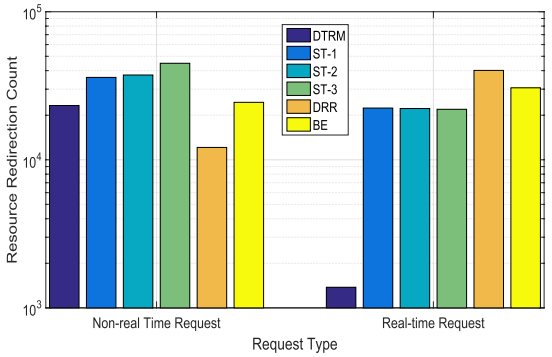
(b) Resource usage

FIGURE 6. Performance comparisons of different schemes with 256 clients and 16 fog nodes.

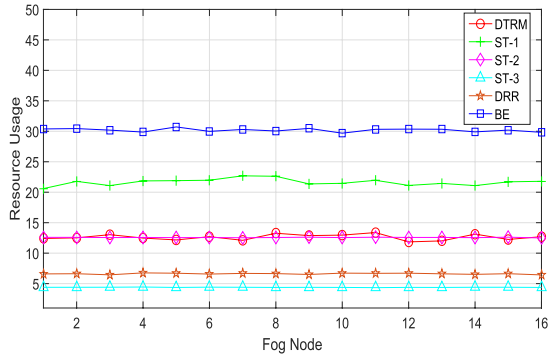
the proportion of RT requests is lower (θ changes from 0.75 to 0.60). The resource usage of the DTRM scheme is comparable to that of ST-3, while outperforming ST-1, ST-2, DRR, and BE. In terms of fair resource allocation, DTRM demonstrates performance consistent with that observed in the previous scenario. These results indicate that DTRM maintains robust and efficient operation under varying numbers of clients and different values of θ, highlighting its adaptability and scalability.

In the third experiment, we evaluated a scenario with 1024 clients and 16 fog nodes. The parameter settings were adjusted as follows: $\theta = 0.50$, on_off_pb = 50, off_on_pb = 3150. Each fog node possessed identical resources to fog node i, specifically, $C_i = 60000$ (MIPS), $M_i = 30000$ (MB) and $B_i = 30000$ (MB) and then $R_i = 60$. This scenario featured the highest request burstiness, with resource availability doubled compared to previous scenarios. The ratio of generated RT requests decreased from 0.60 to 0.50. The experimental results are depicted in Fig. 7(a) and Fig. 7(b). As shown in Fig. 7(a), all schemes consistently exhibited higher resource redirection counts due to the increased burstiness. The DTRM demonstrated the lowest resource redirection count for both RT and NRT requests. For RT requests, the resource redirection count of ST-3 was better than ST-1, ST-2, DRR and BE but significantly higher than that of the DTRM. For NRT requests, BE exhibited a lower resource redirection count than ST-1, ST-2, and ST-3 due to the absence of constraints on NRT requests. DRR exhibits the highest resource redirection count for RT requests among all evaluated schemes, while maintaining a relatively lower redirection count for NRT requests. As shown in Fig. 7(b), the resource usage of all schemes increased when the available resources at each fog node were doubled from 30 to 60. This indicates that resource usage scaled approximately proportionally with the amount of available resources; for instance, the resource usage of DTRM nearly doubled. Despite the aggressive burstiness of the request pattern and variations in total resource availability, DTRM consistently maintained both low and fair resource usage, demonstrating its robustness under dynamic conditions.

In the fourth experiment, we evaluated a scenario with 128 clients and 16 fog nodes, adjusting the parameter settings as follows: $\theta = 0.80$, on_off_pb = 200, off_on_pb = 800 and $R_i = 60$. The experimental results are illustrated in Fig. 8(a) and Fig. 8(b). As illustrated in Fig. 8(a), the DTRM achieves the lowest resource redirection count for both RT and NRT requests. In contrast, the BE scheme exhibits the highest redirection count for RT requests among all evaluated approaches. However, due to the absence of constraints on NRT requests, BE achieves a lower redirection count for NRT requests compared to ST-1, ST-2, and ST-3. The DRR performs better than BE overall but remains less efficient than the ST variants and DTRM in terms of resource redirection.

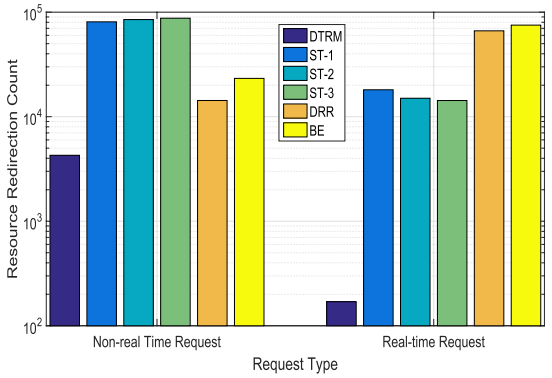


(a) Resource redirection count

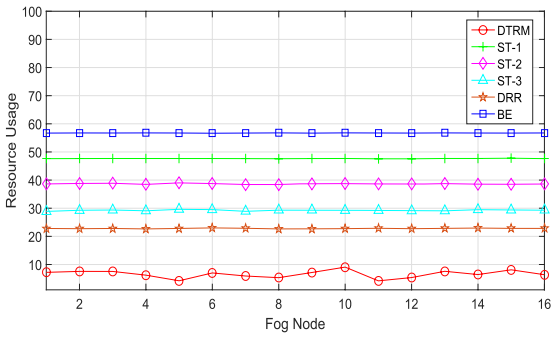


(b) Resource usage

FIGURE 7. Performance comparisons of different schemes with 1024 clients and 16 fog nodes.

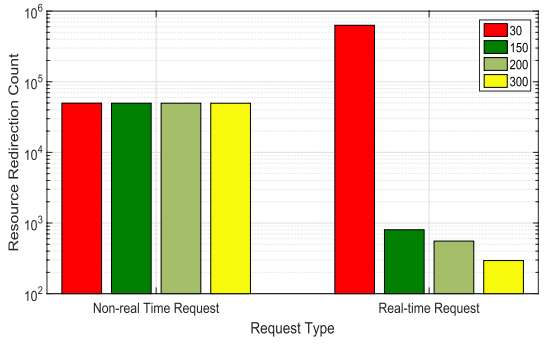


(a) Resource redirection count

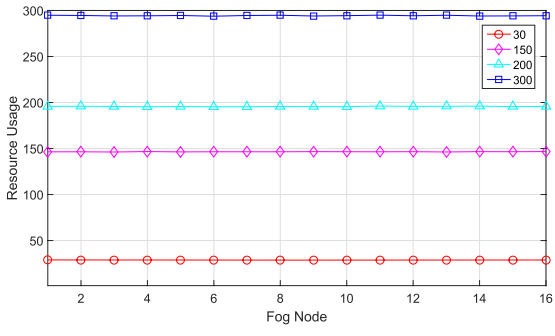


(b) Resource usage

FIGURE 8. Performance comparisons of various parameter settings.



(a) Resource redirection count



(b) Resource usage

FIGURE 9. Performance comparisons of various resource levels in DTRM.

In Fig. 8(b), all schemes exhibit higher resource usage compared to the previous scenario, primarily due to the increase in θ from 0.50 to 0.80. Among them, DTRM maintains the lowest overall resource usage; however, it also displays a larger fluctuation of approximately 5 resource units. This variation is attributed to greater fluctuations in the parameters on_off_pb and off_on_pb. Nevertheless, considering the total of 60 available resource units, this corresponds to only an 8% variation, indicating that DTRM still provides stable performance under more dynamic request patterns.

In the final experiment, we evaluated a scenario with 128 clients and 16 fog nodes, adjusting the parameter settings as follows: $\theta = 0.95$, on_off_pb $= 180$ and off_on_pb $= 20$. This experiment considered the effect of various resource levels in DTRM. The experimental results are illustrated in Fig. 9(a) and Fig. 9(b). In Fig. 9(a), the resource redirection count for RT requests decreased as the total resources increased. Specifically, when the resource units increased from 30 to 150, the resource redirection count for RT requests decreased significantly. However, when the resource units increased from 150 to 300, the decrease in the resource redirection count for RT requests was less pronounced. This indicates that the impact of resource levels diminishes once a certain threshold is reached, suggesting that 150 resource units is a feasible threshold. For NRT requests, which constitute only 5 percent of the generated requests, there was minimal change in the resource redirection count with increased resources. In Fig. 9(b), resource usage was closely aligned with the total available resources, because nearly

all generated requests were RT requests (i.e., $\theta = 0.95$). Additionally, resource usage remained fair despite the aggressive nature of RT requests. Consequently, the DTRM performed well across different resource levels.

The experimental results clearly demonstrate the effectiveness of the DTRM in managing resources within fog computing environments. DTRM consistently achieved the lowest resource redirection counts for both real-time RT and non-real-time NRT requests across a range of scenarios, including variations in client numbers and parameter settings. Moreover, DTRM maintained both low and fair resource usage, outperforming alternative schemes such as ST-1, ST-2, ST-3, DRR and BE. These findings highlight DTRM's suitability for handling dynamic and bursty request patterns involving both RT and NRT requests. Overall, DTRM proves to be a robust and efficient solution for modern IoT applications that demand reliable resource management in fog computing environments.

V. CONCLUSION

Fog computing enhances performance by bringing computational capabilities closer to data sources, thereby reducing latency and alleviating network congestion. Nevertheless, fog nodes are inherently constrained in terms of computing power, memory, and bandwidth. These limitations underscore the need for intelligent resource management strategies that can effectively handle diverse and dynamic workloads. This paper introduced a Dynamic Threshold-Based Resource Management (DTRM) scheme that addresses these challenges by prioritizing real-time (RT) requests and minimizing the redirection of non-real-time (NRT) requests to cloud servers. The DTRM framework incorporates three core components: a resource estimator, a request handling, and a Quota-Based Round Robin (QRR) scheduler. The resource estimator enables accurate prediction of RT request demands, while the dynamic threshold mechanism flexibly adapts to varying workloads. The QRR scheduler ensures fair and efficient resource allocation without requiring per-node state tracking, making it well-suited for scalable fog deployments.

Experimental evaluations under diverse scenarios—varying client numbers, request burstiness, and resource availability—demonstrate that DTRM significantly outperforms existing schemes such as Static Threshold (ST), Deficit Round Robin (DRR), and Best Effort (BE). DTRM consistently achieves lower resource redirection counts, maintains efficient resource usage, and supports balanced load distribution across fog nodes. These characteristics make DTRM a robust and adaptive solution for modern IoT applications with heterogeneous and latency-sensitive service demands. Future work will focus on evaluating additional performance metrics such as end-to-end latency, cost efficiency, and energy consumption. Furthermore, integrating learning-based prediction mechanisms—such as reinforcement learning and long short-term memory (LSTM) networks—could further enhance DTRM's adaptability and precision in highly dynamic environments.

REFERENCES

[1] M. Ghobaei-Arani, M. Shamsi, and A. A. Rahmanian, "An efficient approach for improving virtual machine placement in cloud computing environment," *J. Experim. Theor. Artif. Intell.*, vol. 29, no. 6, pp. 1149–1171, Apr. 2017.

[2] A. Souri, P. Asghari, and R. Rezaei, "Software as a service based CRM providers in the cloud computing: Challenges and technical issues," *J. Service Sci. Res.*, vol. 9, no. 2, pp. 219–237, Dec. 2017.

[3] M. Ghobaei-Arani, A. Rahmanian, M. Shamsi, and A. R. Kenari, "A learning-based approach for virtual machine placement in cloud data centers," *Int. J. Commun. Syst.*, vol. 31, no. 8, Feb. 2018, Art. no. e3537.

[4] A. M. Manasrah, A. Aldomi, and B. B. Gupta, "An optimized service broker routing policy based on differential evolution algorithm in fog/cloud environment," *Cluster Comput.*, vol. 22, no. S1, pp. 1639–1653, Dec. 2017.

[5] N. Wang, B. Varghese, M. Matthaiou, and D. S. Nikolopoulos, "ENORM: A framework for edge NOde resource management," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1086–1099, Nov. 2020.

[6] H. J. de Moura Costa, C. A. da Costa, R. da Rosa Righi, and R. S. Antunes, "Fog computing in health: A systematic literature review," *Health Technol.*, vol. 10, no. 5, pp. 1025–1044, May 2020.

[7] F. Pisani, F. M. C. de Oliveira, E. S. Gama, R. Immich, L. F. Bittencourt, and E. Borin, "Fog computing on constrained devices: Paving the way for the future IoT," in *Advances Edge Computing: Massive Parallel Processing Applications*, vol. 35, 2020. [Online]. Available: https://www.semanticscholar.org/paper/Fog-Computing-on-Constrained-Devices%3A-Paving-the-Pisani-Oliveira/b1dcaa6d4d72d78aff062c01d69880993234777c

[8] C.-H. Hong and B. Varghese, "Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–37, Sep. 2019.

[9] G. Ortiz, M. Zouai, O. Kazar, A. Garcia-De-Prado, and J. Boubeta-Puig, "Atmosphere: Context and situational-aware collaborative IoT architecture for edge-fog-cloud computing," *Comput. Standards Interface*, vol. 79, Jan. 2022, Art. no. 103550.

[10] B. Alturki, S. Reiff-Marganiec, C. Perera, and S. De, "Exploring the effectiveness of service decomposition in fog computing architecture for the Internet of Things," *IEEE Trans. Sustain. Comput.*, vol. 7, no. 2, pp. 299–312, Apr. 2022.

[11] G. Lee, W. Saad, and M. Bennis, "An online optimization framework for distributed fog network formation with minimal latency," *IEEE Trans. Wireless Commun.*, vol. 18, no. 4, pp. 2244–2258, Apr. 2019.

[12] D. Cui, Z. Peng, J. Xiong, B. Xu, and W. Lin, "A reinforcement learning-based mixed job scheduler scheme for grid or IaaS cloud," *IEEE Trans. Cloud Comput.*, vol. 8, no. 4, pp. 1030–1039, Oct. 2020.

[13] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 712–725, Sep. 2019.

[14] 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.

[15] X. Masip-Bruin, E. Marin-Tordera, A. Jukan, and G.-J. Ren, "Managing resources continuity from the edge to the cloud: Architecture and performance," *Future Gener. Comput. Syst.*, vol. 79, pp. 777–785, Feb. 2018.

[16] K. Toczé and S. Nadjm-Tehrani, "A taxonomy for management and optimization of multiple resources in edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, no. 1, pp. 1–23, Jun. 2018.

[17] R. K. Naha, S. Garg, D. Georgakopoulos, P. P. Jayaraman, L. Gao, Y. Xiang, and R. Ranjan, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018.

[18] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. 16th Int. Conf. Service-Oriented Comput. (ICSOC)*, Zhejiang, China, Jan. 2018, pp. 230–245.

[19] M. H. Kashani and E. Mahdipour, "Load balancing algorithms in fog computing," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1505–1521, Mar. 2023.

[20] C. Xu, Y. Wang, Z. Zhou, B. Gu, V. Frascolla, and S. Mumtaz, "A low-latency and massive-connectivity vehicular fog computing framework for 5G," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Abu Dhabi, United Arab Emirates, Dec. 2018, pp. 1–6.

[21] L. Pu, X. Chen, J. Xu, and X. Fu, "D2D fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3887–3901, Dec. 2016.
[22] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Joint container placement and task provisioning in dynamic fog computing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10028–10040, Dec. 2019.
[23] L. Wang, L. Jiao, J. Li, and M. Mühlhäuser, "Online resource allocation for arbitrary user mobility in distributed edge clouds," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 1281–1290.
[24] Y. Jie, M. Li, C. Guo, and L. Chen, "Game-theoretic online resource allocation scheme on fog computing for mobile multimedia users," *China Commun.*, vol. 16, no. 3, pp. 22–31, Mar. 2019.
[25] C. Zhu, J. Tao, G. Pastor, Y. Xiao, Y. Ji, Q. Zhou, Y. Li, and A. Ylä-Jääski, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.
[26] M. Sudhakara, K. D. Kumar, R. K. Poluru, R. L. Kumar, and S. B. Bhushan, "Towards efficient resource management in fog computing: A survey and future directions," in *Architecture and Security Issues in Fog Computing Applications*. IGI Global, Sep. 2019, pp. 158–182, doi: 10.4018/978-1-7998-0194-8.ch010.
[27] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," *Comput. Commun.*, vol. 161, pp. 109–131, Sep. 2020.
[28] L. Gu, D. Zeng, S. Guo, A. Barnawi, and Y. Xiang, "Cost efficient resource management in fog computing supported medical cyber-physical system," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 1, pp. 108–119, Jan. 2017.
[29] N. Abu-Amssimir and A. Al-Haj, "A QoS-aware resource management scheme over fog computing infrastructures in IoT systems," *Multimedia Tools Appl.*, vol. 82, no. 18, pp. 28281–28300, Feb. 2023.
[30] H. K. Apat, B. Sahoo, and S. Mohanty, "A quality of Service(QoS) aware fog computing model for intelligent (IoT) applications," in *Proc. 19th OITS Int. Conf. Inf. Technol. (OCIT)*, Bhubaneswar, India, Dec. 2021, pp. 267–272.
[31] T. Wang, J. Zhou, A. Liu, M. Z. A. Bhuiyan, G. Wang, and W. Jia, "Fog-based computing and storage offloading for data synchronization in IoT," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4272–4282, Jun. 2019.
[32] X. Xu, S. Fu, Q. Cai, W. Tian, W. Liu, W. Dou, X. Sun, and A. X. Liu, "Dynamic resource allocation for load balancing in fog environment," *Wireless Commun. Mobile Comput.*, vol. 2018, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1155/2018/6421607
[33] S. Agarwal, S. Yadav, and A. K. Yadav, "An efficient architecture and algorithm for resource provisioning in fog computing," *Int. J. Inf. Eng. Electron. Bus.*, vol. 8, no. 1, pp. 48–61, Jan. 2016.
[34] A. Buchade and M. S. Nirav, "Priority based allocation in cloud computing," *Int. J. Eng. Res. Technol.*, vol. 3, pp. 855–857, Jan. 2014.
[35] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," *IEEE/ACM Trans. Netw.*, vol. 4, no. 3, pp. 375–385, Jun. 1996.

JUI-PIN YANG (Member, IEEE) was born in Kaohsiung, Taiwan, in 1972. He received the Ph.D. degree in electrical engineering from National Chung Cheng University, in 2003. From 2004 to 2008, he was an Engineer and a Project Leader of the Industrial Technology Research Institute (ITRI). In 2013, he was a Visiting Scholar with the National University of Singapore and Hokkaido University. He is currently a Professor with the College of Marine Resources and Engineering, National Penghu University of Science and Technology, Taiwan. From 2010 to 2018, he was recognized as a Special Outstanding Researcher by the Ministry of Science and Technology, Taiwan. His research interests include computer networks, the Internet of Things, artificial intelligence, and information systems.