ORIGINAL RESEARCH PAPER
Pages: 283-302

An Optimized Method for Outsourcing and Computational Offloading in Resources Allocation to IoT Users in Fog computing

[1]Mahmood Lakzaei, [2]Vahid Sattari-Naeini and [1]Amir Sabbagh-Mollahossini
[1]Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran.
[2]Computer Engineering Dep., Shahid Bahonar University of Kerman, Kerman, Iran
mahmood.lakzaei@iauk.ac.ir, vsnaeini@uk.ac.ir, sabbagh@iauk.ac.ir
Corresponding author: *vsnaeini@uk.ac.ir*

DOI: 10.22070/jce.2021.14497.1187

***Abstract*- Fog computing is a method for improving cloud computations performance attempts to expand the Internet of Things processes and distribute cloud services in the network edge. This paper proposes a real-time outsourcing and offloading mechanism to optimize the cache and CPU consumption in resource allocation to IoT users in a fog-based processing environment. Based on this mechanism, the computations that require heavy processing are moved to the network edge, and computations with lower processing needs are processed inside user devices. According to the simulation results, the average users' average service latency in the proposed method SPA-(Offloading) for 200 users has been improved in the range of 0.8 to 0.6. In addition, the profits of cloud and fog service providers for 220 users are higher than other methods. Also, the average system cost performance is evaluated, which is better than the other methods. The results show that this mechanism improves cache consumption, processing time, and optimal resource allocation to IoT users.**

***Index Terms*-** Cache, Fog computing, Internet of things, Student project allocation algorithm.

I. INTRODUCTION

Fog computing service delays, it generates significant traffic for the Internet. In the face of these challenges, patterns of responding to latency-sensitive needs in the Internet of Things using cloud computing are shifting to the distributed computing approach [3]-[4]. Also, limited device resources at the edge often create problems for delay-sensitive applications. Therefore, some processing tasks must be sent to cloud centers. But since there are so many devices on edge, a good method to select these tasks and offloading them in the cloud should be used. Applying the computational offloading process can solve limited resources at the edge, especially for computationally heavy tasks. This

Manuscript received 7-Sept.-2020 and revised 16-Oct.-2020,
Accepted on 28- Nov.-2020
P- ISSN: 2322-4088
E- ISSN: 2322-3936

mechanism helps to improve efficiency and reduce energy consumption. In this paper, the issue of mutual benefit between users and service providers is stated. The proposed approach can be an optimal framework for better resources allocation based on the list of priorities which leads to more interaction between users and service providers. Due to the limitations of similar methods such as the Hungarian method, which is used for resources allocation and provides the optimal answer with the minimum criterias, so to get the optimal answer with the maximum criterias; In this method, it is necessary to make changes in the main problem and its variables. While, in the student project allocation method, to obtain the optimal answer, there is less need to make changes in the problem. Therefore, due to the importance of stability in the main variables in this article, the student project allocation method has been used. The SPA method can maximize profits by provided a list of priorities and limiting that list, adapting the student to the project, making preference lists in case of a change in decisions, and using real methods instead of statistical and random methods in the resource allocation process. This method has been proposed with the aim of increasing the profit and quality of services to users, user interest rate, average system cost, and reducing latency. Due to the importance of the offloading process in the network resources allocation, synthesize SPA with Offloading, leads to increased response speed to users, efficiency, and quality of services. The simulation results show the optimal performance of this method in the radio and computational resources allocation to IoT users, compared to other similar methods. In this method, the average SP profit is about 4.5%, and the average performance of the system cost for 210 users is about 47% better than the best of other methods. Also, users' average service latency in the proposed method for 200 users is 0.6 sec. which is 25% lower than the best of other methods. The present paper is organized as follows: the literature review is discussed in Section II. The problem statement, context, and principles for the joint resource allocation scheme are presented in Section III. Afterward, in Section IV, the proposed methodology and SPA adaption approach for system modeling and SPA-(s,p) algorithm as a distributed solution are assessed. Simulation results are analyzed and evaluated in Sections IV, V, and VI. Finally, section VII concludes the paper.

II. LITERATURE REVIEW

By applying fog computing, service providers can exchange shared radio and computational resources between objects. For such applications, for example, the fog computing perception mechanism of IoT in [5] can be considered. This mechanism has adequate and flexible node data properties that can improve IoT data resources effectively. In the medical field this developed system can be used in IoT-based diagnosis in health care systems, where accuracy and real-time diagnosis are essential [6].

In [7], a multi-user storage system has been proposed so that users actively run on cache at their request. Caching and embedding are jointly configured in FDMA (Frequency-Division Multiple

Access) settings to reduce the edge server's energy usage and consumers under computing limits, communications, and storage capacities, as well as limitations.

In [8], to minimize the overhead of the fog computing network, including the task process delay and energy consumption, proposed a QoS-aware resource allocation scheme, which jointly considers the association between fog nodes (FNs) and IDs. The simulation results demonstrate that the proposed scheme could efficiently ensure the loading balance of the network, and reduce the network overhead.

In [9], According to wireless transfer technology and information concurrency, the joint discharge of tasks and energy in IoT networks fog has been surveyed and It has been indicated that when there are queues of tasks within the nodes, each task's evacuation decision is made temporarily. Then the optimal strategies for evacuating tasks and energy are mutually identified over several periods. In this article in order to jointly minimize the task execution delay and the energy consumption, an algorithm is proposed.

In [10], a cloudlet-based RL optimization system SDEC (Software-Defined Edge Computing) using new tool has been proposed. This approach demonstrates how ML can address the offloading problem and resource allocation in MEC (Multi-access edge computing) networks. This method provides services for all users in SDN-based (Software-Defined Networking) edge networks by sharing experiences in computing tasks by all service providers.

In [11], a dynamic optimization scheme for an IoT fog computing system is proposed. This scheme is aimed at minimizing the system cost associated with delay and energy consumption. This scheme includes an algorithm for offloading and computing shared radio resources based on the Lyapunov optimization method. The proposed algorithm minimizes the upper limit obtained from the system performance and solves the main problem by dividing it into several parts in each time interval.

In [12], the goal is to minimize the average energy consumption by guaranteeing stability for all system queues. The PORA (Predictive Offloading and Resource Allocation approach) is a distributed offloading and resource allocation scheme for multi-layer fog computing systems that takes advantage of expected offloading to reduce energy consumption while maintaining queue stability. Simulation results show that, this method cause near-optimal power consumptions.

In [13], a scheme is proposed for offloading and computational resources allocation with energy-saving to minimize the system performance cost. The topic of computing task loading and fog computational resource allocation in IoE (Internet of Everything) is discussed in this scheme. The problem is formulated to reduce device costs. Simulation findings show that the proposed plan will save up to 50% on device costs instead of current systems.

In [14], an efficient load balance technique (ELBS) is used for real-time fog computing via fuzzy and probabilistic neural networks, fog computing for supporting real-time programs such as healthcare, industrial systems, and intelligent traffic signs. This is a suitable strategy to achieve load balancing in fog environment as it guarantees a reliable execution for real-time applications. Results showed that the method outperforms load balancing similar methods as it achieves the lowest average turnaround time and failure rate.

In [15], was introduced peer-to-peer (p2p) mechanism considering the cloud computation limits in terms of bandwidth, delay, real-time response, fog computing to perform the IoT devices' requests sent to the cloud. This method minimizes the requests that go to the cloud. This proposed (p2p) model enhances fog computing by adding a p2p mechanism into the fog layer, which allows the fog nodes to collaborate in order to meet the client's needs. The results show that it has better outcomes in terms of bandwidth throughput compared to cloud computing and fog computing models.

In [16], a shared cache mechanism with a communication mechanism is proposed, which involves software fetching, hiding and multicasting, loading task input data, job execution, and computational loading results. This mechanism minimizes loading and time allocation policy through optimization and pooled storage and reduces energy consumption, considering storage and time constraints.

In [17], it is proposed that cache-conscious be replaced to decrease hybrid main memory cache costs, reducing the amount of memory caching in NVM (Non-Volatile Memory) and improving hybrid memory cache capacity. This study's experimental findings illustrate that this conscious cache performs better in system performance and enhances the performance up to 43.6% (on average, 15.5%) than before.

In [18], sub-channel allocation and power control to maximize the total rate at the two-cell network are investigated. Assuming there are some sub-channels in each cell that should be assigned to some users. The proposed method solves the problem of each user's power and the problem of sub-channel allocation, by adjusting the power consumption and assuming the area of low channel interference, according to the Hungarian method. The proposed algorithm was able to formulate the power consumption of each user that the numerical results show the better performance of the proposed method in the signal range on lower noise compared to other previous methods.

III. SYSTEM MODEL

In this section, first, some basic parameters such as outsourcing issue, modeling of cache, and processing time are explained, and then the main system model is presented as follows:

A. Outsourcing Issue

Process outsourcing consists of three main parts: the application, preparation, and decision-making programs to outsourcing. Consequently, computation outsourcing can be moved from IoT devices to the remote cloud server. The first step is partitioning the application programs for outsourcing the necessary computations since this section divides the application program into outsourceable and non-

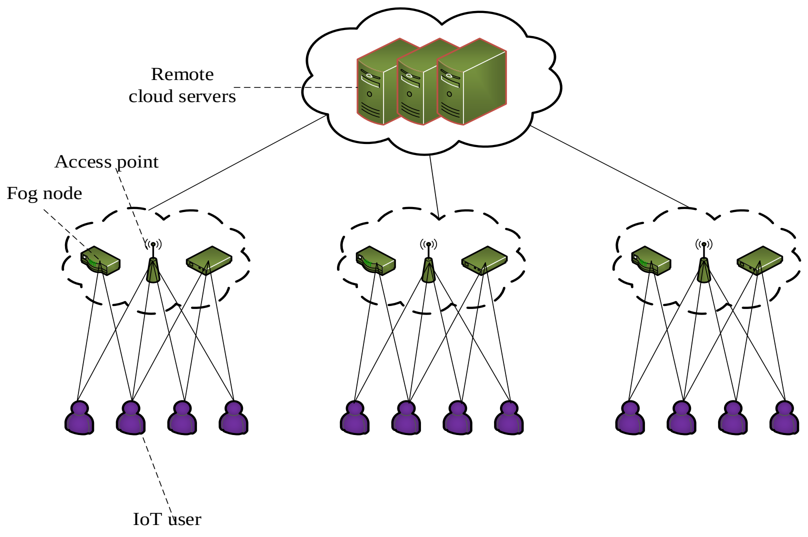

Fig. 1. Outsourcing IoT systems with users, fog nodes, and remote cloud servers

outsourceable computations. It determines which component needs to remain on the IoT device or fog server and what part be moved to the cloud server. However, deciding whether an element be outsourceable or not depends on various data. The programmer can label program components. For instance, he may outsource only some APIs. Also, outsourcing can be based on the program's processing time or its required memory during the runtime. The preparation part consists of all the activities necessary for outsourceable components, which involves selecting all remote servers, transforming and installing the code from small IoT devices. Therefore, to solve this issue, outsourcing decisions are the final step before executing on the remote server, which is performed for outsourced components. If outsourcing decision is during the runtime, more accurate information needs to be accessed [19].

We are thinking about an IoT system with a hierarchical calculation structure and a collection of IoT members, fog nodes, and cloud remote servers. The tasks may be performed locally by each IoT customer or outsourced to computer servers. Fog nodes and external cloud networks are computer servers. In their vicinity, Fog Nodes can provide IoT users with computer services. IoT users can outsource their computing tasks to the fog nodes. IoT outsourcing scheme is shown in Fig.1.

Computations that require heavy processing are outsourced towards network edge via real-time outsourcing and computations that need low processing are processed in IoT and user devices. This model consists of three steps, which are, respectively, as follows:

- In the first step, the time needed for processing is calculated. By considering the request type (heavy and light requests), if they need a lower amount of time, they are processed inside the IoT device, and if heavy processing is required, they are sent to the fog server queue, and the mean waiting time for new requests is calculated. Overall, when a request enters the fog, it is queued until the fog is prepared for proposing it. Then we can obtain the total delay by computing the queue wait and request processing times.

- Second step, the best time for fog node is found based on a cost dependent. This step lists the best neighbor nodes that can handle the additional load and process the IoT request. This list is created based on the position of the nodes (i.e., the closest fog).
- The third step involves maintaining the fog node. When the queue request is beyond fog node capability, outsourcing to a cloud server or another edge is performed.

B. Processing Time Modeling

A cloud edge computing model, comprises several remote servers, bases, edge servers, and limitless IoT or bus terminals. Near each base station is a cloud service side. The base station is accessed from a bus terminal or IoT system through a wireless network. A high-bandwidth cellular network is used to link the base station and the edge server. Therefore, we presume that the edge of the server is at the base station and that the device fog nodes are in the form {1,..., S}. Further, we take the range of S={0, 1, ..., S}, which shows all remote cloud service servers and is utilized in modeling all remote cloud servers into consideration. IoT users may then use them to upload their computer activities.

We believe there is enough computational power on remote cloud servers. Each cloud server is considered an exclusive F computing power virtual machine, which defines server availability per unit of time. In contrast to remote cloud servers, the computing capacity of fog nodes is reduced. Each fog node is evenly distributed between IoT users. Fog nodes cannot prioritize IoT users' services since they take their priorities into account and follow strategic behavior. Therefore, several IoT devices, including smartphones and monitoring cameras, mobile phone cameras, fire alarms, etc., seen as the users U={u1,u2,u3,...} is presumed to exist. Each IoT user can receive a specific cloud service provider (SP) computer or storage service, showing it to be sp = {sp1,sp2,...}. These SPs can observe various users with particular computational needs regarding data sizes and service delays.

The processing time of each user's device is shown as T= {t1, t2, t3, …}. Computational tasks are sent to the cloud for those users that are not delay-sensitive. On the other hand, if the users are delay-sensitive, SPs assign one of the nearby fog nodes (FNs) to outsource computational tasks. FNs closer to users leads to lower transmission delay. CPU computing time is determined by the processing rate of each fog node (FN). Each spj allocates radio resources of $W^j = \left\{w_1^j, w_2^j, \ldots, w_L^j\right\}$ (channel bandwidth) and computing resources $C^j = \left\{c_1^j, c_2^j, \ldots, c_L^j\right\}$ (CPU cycle rate) with a processing time $T^j = \left\{t_1^j, t_2^j, \ldots, t_L^j\right\}$ while selecting the proper FN from $FN^j = \left\{fn_1^j, fn_2^j, \ldots, fn_L^j\right\}$ each user's settings. From the viewpoint of users who process delay-sensitive content, they propose value and priority to SPs for better resource competition. Therefore, here we consider a threshold time t, and each user with a time higher than the threshold outsources the computations to fog edge. Users with processing time lower that threshold utilize a lower CPU clock frequency, and therefore we propose a competitive value for acceptance. Outsourcing time (T_{app}) between IoT devices and remove server is computed according to the following relation [20]:

$$T_{app} = \frac{d_t}{BW} + t_{ex} \tag{1}$$

Parameters such as outsourced data (d_t), network bandwidth, and outsourced program runtime are in the server (tex). IoT devices can estimate the connection time from the available program and communication channel to send the computation data while outsourced. It receives the data of the computing results from the IoT device server. Computing runtime in the server (t_{ex}) depends on various factors such as CPU frequency, communication bus frequency, cache size, number of active cores, and properties of outsourced programs. Hence, t_{ex} is calculated using the below method:

$$t_{ex} = \frac{c_{cpu}}{f_{cpu}} + \frac{c_{bus}}{f_{bus}} + \frac{c_{io}}{f_{io}} + \frac{c_{mem}}{f_{mem}} \tag{2}$$

Parameters C_{cpu}, C_{bus}, C_{mem}, and C_{io} are constant and depend on instructions, and f_{cpu}, f_{bus}, $f_{io,}$ and f_{mem} represent CPU core, bus, input-output, and memory, respectively.

C. Cache Memory Modeling

Cloud computing makes it possible to store and access data and programs over the Internet without requiring local data storage. The computing performance in cloud computing can be improved using cache. In the caching process, the recent internet content and requests are stored. Once the data is stored, when requests for data are received, these data are not searched on the web, but a load balancer responds to these requests. It reduces the time it takes to respond to requests and reduces the workload on the web server. The cache is a service operator that locally stores recent requests and works content. The load balancer responds to requests similar to those stored in the cache instead of being sent to the web server. As a result, request time for such requests is shorter than usual and reduces server workload. Assume that $U = \{u_1, u_2, \ldots, u_k\}$ are upper limits of cache capacity of server data and $C = \{c_1, c_2, \ldots, c_k\}$ are requests in the server queue. For each data content j, x_{jk} is examined to show whether data are hidden in the server edge k[21].

$$x_{jk} \begin{cases} 0, \; k \text{ does not cache data } j \\ 1, \; k \text{ caches data content } j \end{cases} \tag{3}$$

The server determines whether to store data content on the server's side. Depending on the data records used, a cache control mechanism is employed. It ensures that if cache memory is available, it will store as much data as possible. If cache memory is complete, utilization records are used to find the appropriate cache memory for replacement. Since the controller analyzes the real-time cached

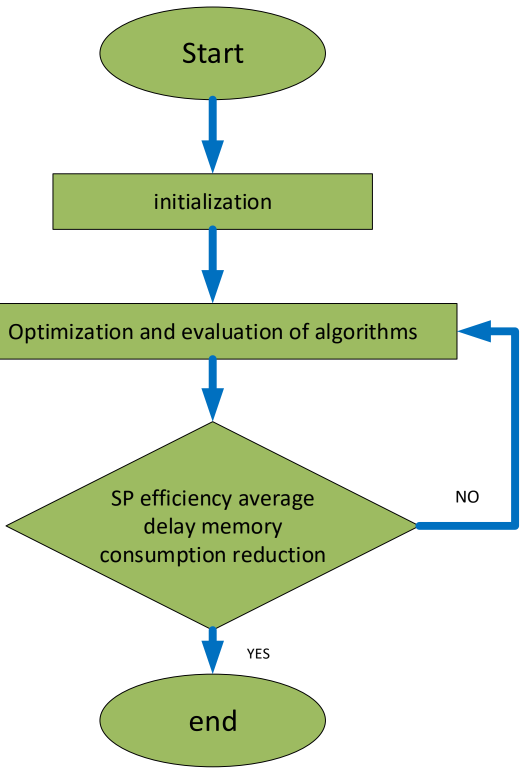
Start
initialization
Optimization and evaluation of algorithms
SP efficiency average delay memory consumption reduction
NO
YES
end

Fig. 2. Outsourced cache management flowchart.

block, the cache controller can be accessed at any time, and the type of data stored in the edge server can be investigated. The flowchart and mechanism for cache management are shown in Fig. 2.
Based on records for storage utilization and outsourcing, a cache controller changes and decides the cache policy. The y_{ijk} function is utilized to determine whether or not the terminal computer is processing the computational work of j in edge server k. When k=0, the terminal computer delegated computing operations to a cloud computing center in another location.

$$y_{ijk} \begin{cases} 0, device\ i\ \ performs\ \ job\ \ j\ \ locally \\ 1, device\ i\ \ offloading\ \ job\ \ j\ at\ k \end{cases} \tag{4}$$

If a terminal computer sends working data j to an edge server, the transmission delay T^{j}_{ik} between edge servers i and k can be determined T^{j}_{ik} using the following form:

$$T^{j}_{ik} = e_{ik} * d_{j} \tag{5}$$

The e_{ik} denotes the time it takes for data to travel from terminal unit i to server edge k. During work computation j between edge servers, d_j shows the data size that needs to be transmitted between the edge and local CPU. When k=0, e_{ik} displays the unit data transfer delay between the terminal computer and the remote cloud server and the size of the data to be sent. Furthermore, C^{j}_{k} is the time spent running task j in the edge server k, including waiting and computing time. Additionally, the waiting period refers to the time elapsed by a completed task or nap, while computing time refers to the time elapsed by a working task. *K=0* denotes the time it takes to run task j on the central cloud server. C_k^j denotes the task's local running time as shown by terminal system i.

For equivalence the parameters with time, the energy consumption are converted to the equivalent execution time and the offloading problem is translated to the equivalent response time. Also, if a set of devices in the terminal covered by edge server k is displayed with M_k, then the equivalent weighted response time for all jobs ($J = \{1.2\ldots.j\ldots\}$) in M_k, which must be minimum, is calculated as:

$$P = \sum_{j \grave{o} J} \sum_{i \grave{o} M_k} \left\{ r_j * y_{jik} * \left(C_k^j + \left(1 - x_{jk}\right) * T_{ik}^j \right) + \left(1 - y_{ijk}\right) * \hat{C}_i^j \right\} \tag{6}$$

Where r_j is job j 's delay sensitivity and $\hat{C}_i^j$ is local equivalent weighted execution time which is calculated as follows:

$$\hat{C}_i^j = \beta * \left(\frac{E_j^c}{E_j^t} \right) * r_j * C_i^j \tag{7}$$

where β is the coefficient of local energy consumption, so the larger value of β, shows the more device attention to local energy. C_i^j is the local execution time of job j , done by terminal device i. E_j^c is the required computation energy consumption when performing computation job locally and E_j^t is the transmission energy consumption. If $E_j^c > E_j^t$ then the work is sent to the edge server or cloud server; otherwise, it runs on the device.

IV. PROPOSED METHOD

In this section, while presenting a theoretical framework, first the basic definitions such as student project allocation issue and user satisfaction with service providers and profits of SPs from resources allocation to users are presented; then the main research problem and its algorithms are provided as follows:

A. The student project allocation issue

Student Project Allocation (SPA) problem is a method where several projects are allocated to some students with the help of lecturers. According to the structure and framework of the SPA problem, each lecturer provides students with various projects. Each student can decide about the available projects. They can choose to accept or reject any of these projects. A lecturer can provide several projects, or there may be many students assigned to a project. Figure 3 shows the allocation of radio and computing resources in this study and compares it with the SPA problem. Indeed, the students and projects are considered equivalent to devices on IoT and radio spectrums of fog nodes. Also, the lecturer is regarded as a cloud resource. SPs provide radio resources packets, and existing processors

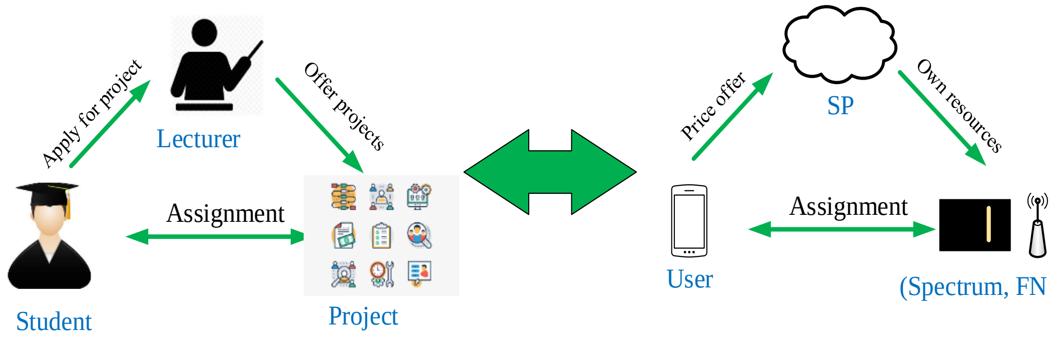

Fig. 3. The student project allocation modeling.

and the users are suggested to SPs for allowed packets. Therefore, it is enough to determine classes for assigning projects to students and then perform simulation.

In this allocation process, it is essential to consider the critical definitions described in the following.

B. *User satisfaction*

One of the most relevant metrics for assessing service efficiency is user satisfaction. Service delay can be used as a criterion to measure user satisfaction if a set of delay-sensitive users are involved. As the first step, it should be ensured that the quality of exchange between users and FNs (Fog nodes) satisfies such a need. In other words, to have accurate and complete results, the Signal to Interference and Noise Ratio must be greater than the threshold. The SINR (Signal to Interference *and* Noise Ratio) obtained from in use can be defined as:

$$\Gamma_{k,1}^{i,j} = \frac{P_i g_{k,1}^{i,j}}{\sum_{u_{i'},\in u,i'\neq i} \rho_{k,1}^{i',j} \mathrm{P}_{i'} h_{k,1}^{i',j} + \sigma_N^2} \tag{8}$$

where P_i and $g_{k,11}^{i,j}$ are transmission power and channel increase between user $\mathrm{U_i}$ and fog node $\mathrm{fn_1^j}$ using channel W_k^j, respectively. $h_{k,1}^{i',j}$ denotes the profit of interference channel obtained from each of the other mobile users $\mathrm{U_i}$ in $\mathrm{fn_1^j}$ due to channel reusing. Here, we assume that radio resources are used among SPs, and radio resources in SP can be coordinated to prevent interference. σ_N^2 denotes channel noise. It is required to have $\Gamma_{k,1}^{i,j} \geq \Gamma_{min}$ for successful transmission. If SINR requirements are met, transmission rate from $\mathrm{U_i}$ in $\mathrm{fn_1^j}$ using W_k^j can be written as:

$$r_{k,1}^{i,j} = w_k^j \log\left(1+\Gamma_{k,1}^{i,j}\right) \tag{9}$$

Transmission time, CPU loading time, and receipt time are three-time intervals that can trigger delays in delivering services. As a result, interference from other users on the shared channel will affect each user's transmission rate. Furthermore, the shared CPU running rate with each device is influenced by the shared CPU users. For the sake of convenience, each shared CPU user is assigned an equal share of the overall CPU rate. This allocated share is denoted by $c_{k,1}^{i,j} = \frac{1}{\sum_{u_i \in u} \rho_{k,1}^{i,j}} c_1^j$ ..

Thus, the service delay for the time that the resource pair $\left(W_k^j, C_1^j\right)$ i is used can be defined as follows:

$$t_{k,1}^{i,j} = t_{trans} + t_{proc} + t_{recv} = \frac{D_i}{c_{k,1}^{i,j}} + \frac{DC_i}{c_{k,1}^{i,j}} + \delta t \quad (10$$

In the general process, the users are all allowed to inform service providers about their needs. Also, service providers try to satisfy fog users by establishing relationships with them to find appropriate nodes for users' loading computing tasks and assign the required radio spectrum to meet their needs. Calculating service delay using formula (8) and considering it in the allocation process leads to user satisfaction.

C. SP profit

Mandatory profit is a factor causing SPs to provide better service for their common users. Another factor for evaluating the system performance can be price suggestions from users as SP profit. The price suggested by each user does not depend only on T_i delay; its size D_i affects it. We presume a linear relationship occurs between the price and the size of the data without sacrificing generality, and there is a delay inversion. Thus, the suggestion from each user can be presented as the following.

$$O_i = f\left(D_i, T_i\right) \quad (11)$$

where f must be an incremental function for D_i and a decreasing function for T_i. for simplicity, we use the below function to define $f(D_i, T_i)$.

$$O_i = \frac{aD_i}{T_i} \quad (12)$$

in which, *a* is a parameter with unit dollar/Mbps, and Oi is the price ui tends to pay to each SP in the case of adaptivity. Each SP provides service for more than one user and therefore receives more than one suggestion. SP profit is defined as total offers collected from all users. Each SP's price can be associated with electricity consumed for transmission and storage, which is considered to be fixed for simplicity. We neglect fixed services costs while considering the SP profit. Consequently, the total profit for each SP is defined as follows [22].

$$Reu_i = \sum_{u_i \in u} \rho_{k,1}^{i,j} O_i \quad (13)$$

D. Spa- (S, P) offloading algorithms in research

Unstable matching between resources and users may lead to a problem in which two servers (SPs) exchange data with their identical users simultaneously. Such challenges will eventually lead to undesirable and unstable operations in the network. Therefore, the SPA-AP algorithm can be used as an efficient method to calculation time proceessing and outsourcing between users and resources. Algorithm(1) reflects this approach. According to the cache modeling done in the previous section, in algorithm (2), the cache's extent is used in the resource allocation process is investigated.

- *SPA- (s, p) offloading algorithm of processing time modeling*

Edge devices have limited resources. Therefore, some processing tasks must be sent to the cloud centers. In other words, such tasks are offloaded to the cloud. However, there are many devices on edge. So we need a mechanism to select these tasks to offload them to the cloud. This problem can be modeled and solved using the proposed algorithm.

In this algorithm, we must first calculate the interference caused by offloading the workload of each device. To this end, the devices that follow the task offloading strategy first send a test signal to the base station. The base station receives these test signals on each channel and calculates the interference. The base station then gives feedback to the devices. As a result, each device can detect interference in its chosen channel and decide whether to offload its workload. If a device decides to offload, it will send a request to the cloud to update information. Then, the cloud randomly selects one of the devices that have sent requests. This algorithm consists of a set of variables with two input, output parts. In the meantime, we can start the processing operation by initialization.

- *SPA- (s, p) offloading algorithm of cache modeling*

Cache memory applies complicated algorithms to predict and store CPU-required data for processing.

The cache uses sophisticated algorithms to predict and store the data required for the processing of the processor. Therefore, the CPU first searches for the required data in the cache. If this data is found in the cache, the CPU reads it. This data is stored in a cache that is inside or near the processor. Therefore, in this method, we will achieve a high-speed process. However, in the proposed algorithm, the task must be assigned to the machine with the lowest workload. In this method, the cache alternately examines the queue of requests and assigns them to virtual machines. The cache also maintains a list of assigned tasks to determine which virtual machine should be assigned the new task. In this algorithm, the client requests and selects the appropriate virtual machine. Then the cache finds the suitable virtual machine cache for subsequent requests. According to SPA-AP algorithm features, this method includes two sections; input and output

Algorithm 1: *SPA- (s, p) offloading algorithm of processing time modeling*

Input: Users U, bandwidth W, fog nodes FN, the processing time for each user, threshold time for outsourcing.

Output: Modification M.

Initialization: The list M is empty, and all users are free. First SPA-AP algorithm pseudocode and processing schedules will be defined as the following:

```
1:      Putting the result [ajk] after reaching the desired solution
2:        Initialization : Execution Time = $t_{jk}$;
3:          Put users (ui) in the first location rp_(1,k)^j in ρL_i^user. Add a new user
             to list M and pick this user from the list ρL_i^user.
4:            For all computing resources (rp_(1,k)^j)
5:            While rp_(1,k)^j, if the sum of the times C-cpu,  C_bus,  C_mem, and C_io;
               is lower than the threshold time, perform the following tasks
6:             Find (uwst, rpwst) for outsourcing the rp_(1,k)^j in list spj
7:            Remove (uwst, rpwst) from list
8:           End of the loop while
9:         End of the loop for
10:     End of the operation is evaluated via modification in variable M
```

In this algorithm, resource allocation is possible only if the time obtained for outsourcing is lower than the IoT program's processing time. Thus outsourcing process will be performed, and project allocation is achieved.

.Algorithm 2: *SPA- (s, p) offloading algorithm of cache modeling*

Input: Cache memory and remote server

Output: Modification M. Pseudocode of the second SPA-AP algorithm

Moreover, the cache memory can be defined in the following way in allocating resources to the users.

```
1:      Putting the result [ajk] after achieving the desired solution
2:         Initialization of cache memory and server
3:           Put users (ui) in the first location rp_(1,k)^j in ρL_i^user.
              List M adds a new user. It picks this user from the list ρL_i^user.
4:             (rp_(1,k)^j) is calculated for all resources.
5:             Selecting resource k from list LA (resources available in remote server
            for outsourcing)
6:          Initializing the tasks to be outsourced
7:        Finding the lowest available cache among remote server and IoT devices memory
8:      Allocating rp_(1,k)^j based on the lowest cache memory
9:    Terminating the task with a modification in variable M
```

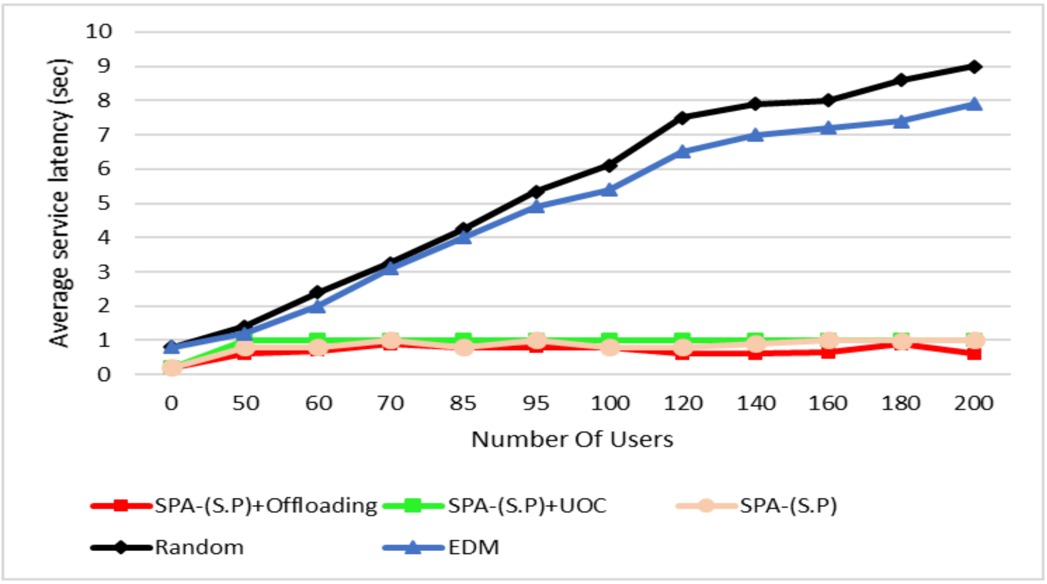

Fig. 4. Users' average service latency

In this algorithm, resource allocation is performed when the cache used in the IoT device is more than the remote server.

V. PERFORMANCE EVALUATION

Variables applied in this SPA(S-P) algorithm include bandwidth, cache memory, CPU cycle, service providers' efficiency (SP), and system cost performance. Some users M are distributed in the network, which each SP has a k=5 channel band for users for sharing, and bandwidth is set to w=5 megahertz. An equal capacity is determined for each channel and fog node, which qr = qc =100 is the service provider capacity. The maximum processing time is assumed as max time proc =10, and users' delay, data size, and CPU cycle are determined according to different IoT devices. In this regard, service consists of two offload and processing delays, which the rate of this processing for each FN as uniform distribution is set to [5 and 6] $*10^{10}$ cycles/second. We define the permittivity constant as $C=10^{-2}$ to increase the constant g propagation.

Fig. 4 shows a compared about average computing and latency between four methods of random method, EDM algorithm, SPA(S-P) algorithm, and suggested SPA offloading process. Overall, service delay increases in all four methods as the number of users increases. However, because a more significant number of users cause lower resource allocation in each averaging, a more considerable delay will result. Fig. 4 shows average service latency for 200 users. This parameter, up to 50 users, is relatively the same for 5 methods. Still with an increasing number of users the average service latency for the three methods SPA-(s, p), UOC and SPA-(s, p) and SPA-(s, p) offloading, have the same values. Still, the proposed method has a lower latency between these three methods than the previous best method by 40%.

Table I. Users' average service latency

Num	Methods	First Latency (sec)	Seconds Latency (sec)	Number of Users
1	SPA-(s,p) ,UOC	1	1	200
2	SPA-(s,p)	0.9	1	200
3	EDM	1.2	0.8	200
4	Random	1.3	0.9	200
5	SPA-(s,p) , Offloading	0.8	0.6	200

Table I and the comparison of listed methods indicate that EDM and random methods have the highest latency lying in the range of 0.8 to 1.2 from 200 users and 0.9 to 1.2 from the same number of users. However, the latency for the SPA-(s,p) Offloading method lies in a normal range and a range from 0.6 to 0.8 from 200 users; it has the lowest latency time compared with other methods. Because of the outsourcing, processing the big data, average runtime, and latency in the suggested scenario decreased significantly compared with other methods.

In the case of medium profits, almost all users can synchronize with one source pair before the number of users reaches the maximum network capacity. However, when the number of users exceeds the network capacity, each user must compete for a share and usually, users who need more time, they offer higher prices and therefore have a better chance of being selected and serviced by SP.

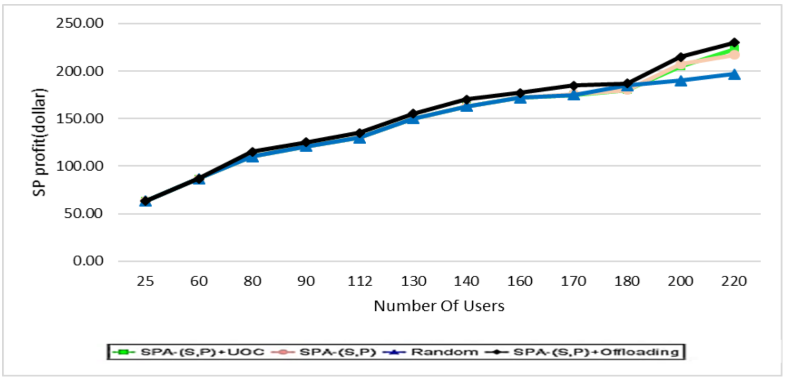

Fig.5. SPs' profit.

As shown in Fig.5, when server profits are up to about 180; users have had similar performance in four methods. But from 180 users up to 220 users, according to the simulation results, the average profit of service providers in the proposed method is 4.5% more than the previous best method. Therefore the service provider can get all the benefits due to the better speed of the proposed method in resources

Table II. The average efficiency of SPs and users

Num	Methods	SP profit(1)	SP profit(2)	Number of Users
1	SPA-(s,p) , UOC	60	220	225
2	SPA-(s,p)	60	210	225
3	Random	60	199	225
4	SPA-(s,p), Offloading	60	230	225

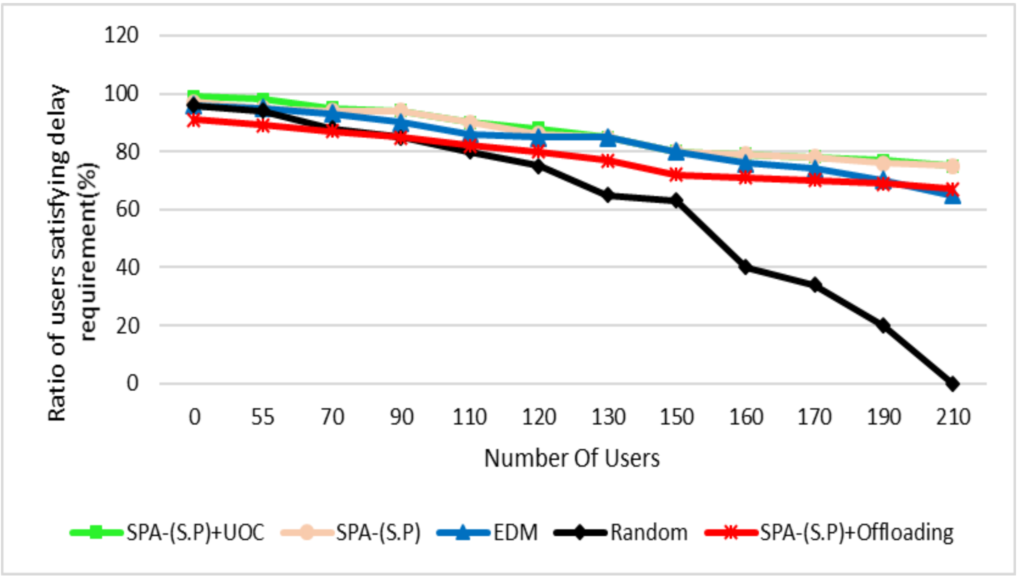

Fig.6. The ratio of users satisfying delay requirement.

Fig. 6 shows the level of user satisfaction with the delay in providing the service. Initially, this item was the same for all four methods, equal to 100%; however, with increasing the number of users, the level of satisfaction, with the delay, decreased to 150 users in 4 other methods, except the random method that was the same and equal to 75%; from 150 users upwards, the variation of this parameter was not significant. Since delay-sensitive users suggest higher prices to service providers, so they are more likely to be selected by SPs, so these users are delayed. On the other hand, users who offer higher price, offers to bring better profits to SPs. Comparing these two items, in Fig. 6 and Fig. 5, shows that the proposed method compared to other methods results in reducing users' latency and increasing SP profits significantly. At the same time, it has better performance due to the optimal management and resources allocation between service providers and users. According to the proposed algorithm, operations that require heavy computing, outsourced and operations with less complex computations are performed on IoT devices. Therefore, the proposed method has less delay than other previous methods.

Table III. Ratio of users satisfying delay requirement(%)

Num	**Methods**	**First Delay (%)**	**Seconds Delay (%)**	**Number of Users**
1	SPA-(s,p) , UOC	99	75	200
2	SPA-(s,p)	97	75	200
3	EDM	96	65	200
4	Random	96	0	200
5	SPA-(s,p) , Offloading	91	67	200

Table IV shows SPA's results-(s,p) performance with offloading in average efficiency rate and system performance.

The evaluation results listed in the tables indicate that in a certain number of users, the efficiency rate and system performance in the SPA-(s,p) offloading method are better than other methods. The proposed method's performance cost is carried out based on the users' rate of profit in allocating the required resources in the suggested algorithm framework.

VI. CONCLUSION

In this paper, the resource allocation problem in fog computing is investigated, emphasizing properties of processing resources in the IoT, using the outsourcing principle to create a reliable relationship between users and resources. SPA-(s,p) offloading modeling is done based on the SPA issue in the proposed system, and an interface is built between IoT users and SPs (service providers) and FNs (fog nodes). The proposed method uses real-time decision-making and can make quick decisions regarding the processing of tasks on IoT devices or fog servers. so a small volume of processing and computational load are performed on the IoT device, while heavier processes and computations are outsourced to the network's edge. As a result, this method achieves a significant improvement in efficiency, cost, and reduced processing time and utilization of system resources such as cache compared to previous methods. This method was also able to improve the average system performance and service latency compared to other previous methods. Improving these parameters finally leads to an increased QoS at the network.

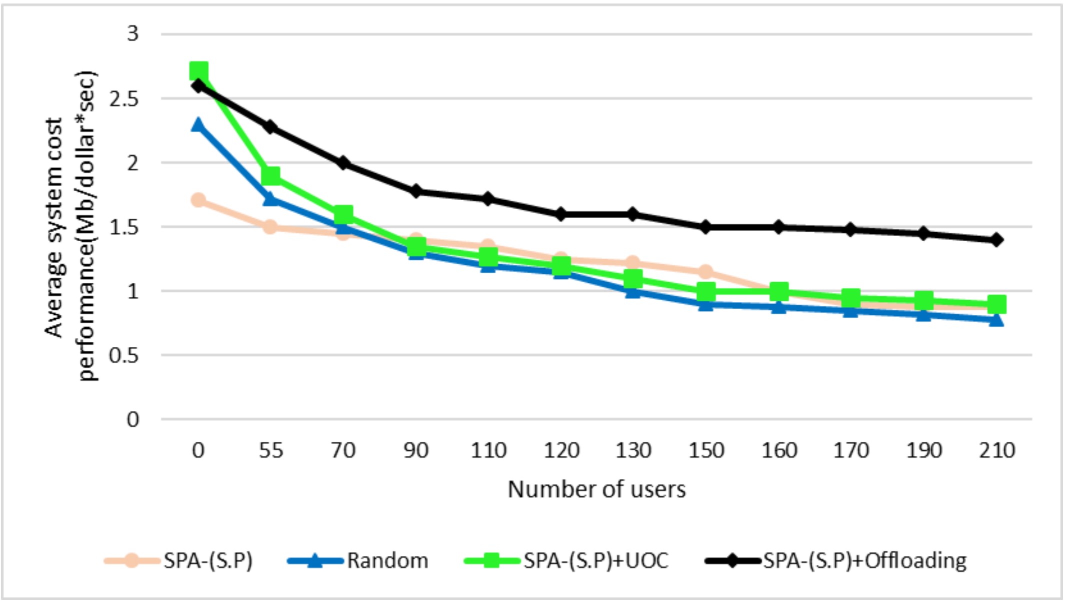

Fig.7. System performance cost in terms of the number of users

Table IV. System performance cost in terms of the number of users

Num	Methods	Cost Performance(1)	Cost Performance(2)	Number of Users
1	SPA-(s,p) , UOC	2.7	0.95	225
2	SPA-(s,p)	1.7	0.95	225
3	Random	2.3	0.8	225
4	SPA-(s,p) , Offloading	2.6	1.4	225

For future work, it is recommended to use the Hungarian method or other similar methods to optimize the resources allocation of IoT users in fog computing.

The results of simulating each of these evaluated methods based on the performance are given in Table III.

Fig. 7 shows the cost of system performance. This parameter is a joint evaluation of the benefits of users and service providers to allocate the best resources to the users who request them the most and pay more for them. The higher the price offered, the more optimal it will be. In this method, all four methods reduce the cost performance of the system by increasing the number of users, because more users will have access to fewer resources. However, the proposed method for this parameter has a 47% better performance than the previous best method. The level is higher than other methods.

REFERENCES

[1] S. M. Mirrezaei "Improving the Efficiency of Wireless Sensor Networks Using Fountain Codes," Journal of Communication Engineering, vol. 9, no. 1, pp. 168-183, Jan. 2020.

[2] J. Tavakoli L. Amini , N. Moghim and F. Pasandideh "A Fuzzy Based Energy Efficient Clustering Routing Protocol in Underwater Sensor Networks," Journal of Communication Engineering, vol. 9, no.1, pp. 154-167, Jan. 2020.

[3] J. Ren, D. Zhang, S. He, Y. Zhang and T. Li, "A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet," *ACM Computing Surveys (CSUR*), vol. 52, no. 6, pp. 1–36, Jan. 2019.

[4] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: from architecture to edge computing and big data processing," The Journal of Super Computing, vol. 75, no. 4, pp. 2070–2105, Nov. 2019.

[5] J. Fei and M. Xiaoping, "Fog computing perception mechanism based on throughput rate constraint in intelligent Internet of Things," Personal and Ubiquitous Computing, vol. 23, no. 3-4, pp. 563–571, July 2019.

[6] J. P. Rajan, S. E. Rajan, R. J. Martis, and B. K. Panigrahi, "Fog Computing Employed Computer-Aided Cancer Classification System Using Deep Neural Network in the Internet of Things Based Healthcare System," Journal of Medical Systems, vol. 44, no. 34, Dec. 2020.

[7] J. Chen, H. Xing, X. Lin, and S. Bi, "Joint Cache Placement and Bandwidth Allocation for FDMA-based Mobile Edge Computing Systems," *IEEE International Conference on* Communications (ICC), July 2020.

[8] X. Huang , Y. Cui , Q. Chen, and J. Zhang, "Joint Task Offloading and QoS-aware Resource Allocation in Fog-enabled Internet of Things Networks," *IEEE Internet of Things Journal*, vol.7, no. 8, pp. 7194-7206, Aug. 2020.

[9] P. Cai, F. Yang, J. Wang, X. Wu, Y. Yang, and X. Luo "JOTE: Joint Offloading of Tasks and Energy in Fog-Enabled IoT Networks," *IEEE 90th Vehicular Technology Conference*, Nov. 2019.

[10] N Kiran, C Pan, S Wang, and C Yin, "Joint Resource Allocation and Computation Offloading in Mobile Edge Computing for SDN based Wireless Networks," *Journal of Communications and Networks*, vol. 22, no.1 pp. 1-11, Feb. 2020.

[11] W. Wen, Y. Cui, T. Q. S. Quek, F. C. Zheng, and S. Jin, "Joint Optimal Software Caching, Computation Offloading and Communications Resource Allocation for Mobile Edge Computing," *IEEE Trans. Vehicular Technology*, arXiv:2005.02627v1, pp. 1-15, May 2020.

[12] X. Gao, Xi Huang, S. Bian, Z. Shao, and Y. Yang, "PORA: Predictive Offloading and Resource Allocation in Dynamic Fog Computing Systems," *IEEE Internet of Things Journal*, vol. 7, no. 1, pp. 72-87, Jan. 2020.

[13] Q. Li, J. Zhao, Y. Gong, and Q. Zhang, "Energy-Efficient Computation Offloading and Resource Allocation in Fog Computing for Internet of Everything," *China Communications*, vol. 16, no. 3, pp. 32-41, Mar. 2019.

[14] F.M.Talaat., S. H. Ali, A.I. Saleh, and H.A. Ali, "Effective Load Balancing Strategy (ELBS) for Real-Time Fog Computing Environment Using Fuzzy and Probabilistic Neural Networks," *Journal of Network and Systems Management*, vol. 27, pp.883-929, Feb. 2019.

[15] E. Schleicher, K. Graffi, and A. Rabay, "Fog Computing with P2P: Enhancing Fog Computing Bandwidth for IoT Scenarios," *International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (Green Com) and IEEE Cyber, Physical and Social Computing (CPS Com) and IEEE Smart Data (Smart Data*), 14-17 July 2019.

[16] Z. Chang ,L. Liu, X. Guo, and Q. Sheng "Dynamic Resource Allocation and Computation Offloading for IoT Fog Computing System," *IEEE Trans. Industrial Informatics*, vol. 17, no. 5, pp. 3348-3357, May 2021.

[17] G. Jia, G. Han, H. Wang and F. Wang, "Cost aware cache replacement policy in shared last-level cache for hybrid memory based fog computing," *Enterprise Information Systems*, vol. 12, no. 4, pp. 435-451, Feb. 2017.

[18] A. Khalili and S. Akhlaghi, “Power Control and Scheduling For Low SNR Region in the Uplink of Two Cell Networks,” *Journal of Communication Engineering*, vol.7, no.1, pp. 34-48, Jan. 2018.

[19] K. Akherfi, M. Gerndt, and H. Harroud, “Mobile cloud computing for computation offloading: Issues and challenges,” *Applied Computing and Informatics*, vol. 14, no. 1, pp. 1-16, Jan. 2018.

[20] R. M. Shukla and A. Muni, “An Efficient Computation Offloading Architecture for the Internet of Things (IoT) Devices,” *2017 14th IEEE Annual Consumer Communications & Networking Conference (CCNC),* July 2017.

[21] H. Wei, H. Luo, Y. Sun and M. S. Obaidat, Life Fellow, “Cache-Aware Computation Offloading in IoT Systems,” *IEEE Systems Journal*, vol. 14, no. 1, pp. 61-72, Mar. 2020.

[22] Y. Gu, Z. Chang, M. Pan, L. Song, and Z. Han, “Joint Radio and Computational Resource Allocation in IoT Fog Computing,” *IEEE Trans. Vehicular Technology*, vol. 67, no. 8, pp.7475-7484, Aug. 2018.