Fog Network Area Management Model for Managing Fog-cloud Resources in IoT Environment

Anwar Alghamdi[1], Ahmed Alzahrani[2], Vijey Thayananthan[3]
Department of Computer Science
King Abdulaziz University
Jeddah, Saudi Arabia

***Abstract*—The Internet of Things (IoT) paradigm is at the forefront of the present and future research activities. The enormous amount of sensing data needing to be processed increases dramatically in volume, variety, and velocity. In response, cloud computing was involved in handling the challenges of collecting, storing, and processing the data. The fog computing technology is a model used to support cloud computing by implementing pre-processing tasks close to the end-user for achieving low latency, less power consumption, and high scalability. However, some resources in fog computing network are not suitable for some tasks, or the number of requests increases outside capacity. So, it is more efficient to reduce sending tasks to the cloud. Perhaps some other fog resources are idle, and it is better to be federated rather than forwarding them to the cloud. This issue affects the fog environment's performance when dealing with large applications or applications sensitive to time processing. This research aims to propose a holistic fog-based resource management model to efficiently discover all the available services placed in resources considering their capabilities, deploy jobs into appropriate resources in the network effectively, and improve the IoT environment's performance. Our proposed model consists of three main components: job scheduling, job placement, and mobile agent software, explained in detail in this paper.**

Keywords*—Resource management; job scheduling; load balancing; mobile agent software; fog computing; Internet of Things (IoT)

I. Introduction

Digital devices have been distributed rapidly in our virtual world. These devices continuously produce a massive amount of structured, semi-structured, or unstructured data such as temperature sensors, health care devices, and transport. The output of these devices and applications results in a considerable amount of process [1]. Most digital devices and applications are connected to the Internet to make our environment smart and provide services anytime and anywhere. Anything that can be connected to the Internet and provide or produce data can be considered as the Internet of Things (IoT), which may reach 75.4 billion things in 2025 [2][3]. The IoT devices have limited processing power and memory availability; therefore, the massive amount of data generated from the sensors is collected in clouds for providing many application accesses and services to the users. However, IoT devices have been rapidly increasing, and the clouds cannot serve all these devices efficiently. Also, some IoT applications need to have processes' results as soon as possible such as controlling the moving vehicles, congestion through a mobile pilot, and medical applications. So, fog computing firstly has been proposed by Cisco in 2012 to address the challenges between IoT devices/sensors and clouds [28]. Fog computing is a modern model which considered an extension of clouds to provide services to network parties [4]. It consists of smaller processing power, smaller memory size, and closer to the end devices. Also, it does some processors before it sends them to a cloud. It can be a significant factor in the success of some applications that are sensitive to time processing when there is a high probability of speeding up emergency detection and warning to support appropriate intelligent decision making [6]. For instance, the author in [5] presents a framework of an early-warning system based on IoT. This kind of system is critical to saving human life by providing a high response warning if there is a flood. Another instance is illustrated in [7]. The face recognition method has been increasing in many fields. It is a significant factor in making security more effective by processing the job accurately and quickly. So, the authors try to conduct the task on fog computing rather than on the cloud side to achieve low bandwidth.

In this paper, we try to solve the problem of when one fog resource is not suitable for a specific task or the number of requests increases outside capacity; it is not efficient to send all tasks to the cloud. Perhaps some other fog resources are idle, and it is better to be federated rather than forwarding them to the cloud, as mentioned in [8]. This issue affects the fog environment's performance when dealing with huge applications or applications that are sensitive to time processing.

This paper aims to provide a new solution that can efficiently utilize the fog computing network's capability and increase the performance of IoT applications. We build a holistic fog-based resource management model which efficiently discovers all the available resources with their capabilities, deploys jobs into appropriate resources in the network effectively, and improve IoT applications' performance by implementing the job locally close to the end-users.

The objectives of this paper are listed as follows:

- Prioritize the jobs according to applications requirement.
- Balance and load the jobs among the fog nodes resources.

- Blend Mobile-Agent in the fog computing environment.
- Track and update the status of the cloud/fog resources.

The following is how the rest of this article is presented. In Section II demonstrates the related work for the relevant methods in the proposed solution. Section III presents the proposed (FNAMM) model. Section IV discuss the proposed model and reveals the benefits and compared to other models. Section V concludes the work and investigates the possibilities for the future work.

II. Related Work

This research's literature review can be classified based on the essential aspects that fulfil the proposed architecture. Initially, the massive amount of data generated from the smart devices would be underlined by considering their IoT environment challenges. Secondly, various studies will be presented covering resource allocation and discovering their specifications. Thirdly, some studies will illustrate the load balancing and selector techniques in the fog environment to achieve high performance.

The IoT devices have been increasing rapidly globally, leading to generating a massive amount of data through different sensors. IoT big data analytics' primary purpose is to enhance business performance by applying processes such as searching a database, analysing, and mining [9]. However, the statistics reveal that there will be around 1 trillion sensors in 2030 [10]. This challenge would be mitigated by providing enormous resources with efficient management.

Cloud computing is a powerful paradigm in providing computation and storage resources for IoT devices. However, the increasing amount of IoT devices leads to high power consumption and high latency; thus, there should be done some process in the edge of the network rather than in cloud computing [11]. Resource allocation and resource scheduling technologies manage the data centres in cloud computing. These technologies enhanced resource utilization and established load balancing for the data centres. As a result, bottlenecks and overloaded have been addressed [12]. Resource allocation is not an easy job in fog computing since the computing nodes are distributed in the network edge. In cloud computing, the computing nodes are distributed in a centralized data centre.

It is not an easy job of discovering edge resources to deploy workloads from IoT devices or clouds [13]. Many techniques are implemented for discovering edge resources using handshaking protocols, programming infrastructure, and message passing. A new handshaking protocols technique for discovering edge resources has been presented in [14]. This technique is based on the Edge-as-a-Service (EaaS) platform, which can discover a set of homogeneous edge resources. This kind of platform needs a master node that can execute a manager process and communicate with edge resources. After identifying the appropriate node, the Docker containers would be deployed on that node. The authors in [15] proposed a new programming infrastructure mechanism called Foglest that allows edge resources to join a cloud system. This mechanism's protocol can match the application's edge resources requirements against the available and appropriate resources on edge.

Moreover, the protocol can select a node from a set of edge resources closer to the user. The last technique for discovering edge resources is message passing. In [16], the user can submit a query to an edge node in the network by relying on simulation-based validation. Nonetheless, the edge nodes are not necessary to be connected to the Internet.

Thus, there is a need for developing resource management for IoT applications to achieve efficient load balancing in the fog environment [17]. Moreover, a system model for managing mobile cloud network's network resources has been presented effectively in [18]. One of the challenges in fog computing is to select appropriate edge resources to place computation tasks from cloud and IoT devices. There is needed for efficient selector algorithms that can address this issue by considering the availability of edge resources with their capabilities [16]. In [19], the authors proposed a new method for managing mobile and edge devices. The fog resources are distributed in decentralized mode, and IoT devices connection is peer-to-peer in a decentralized mode as well. The problem of distributing tasks in fog computing has gained attention from researchers recently. The authors in [20] have analysed the offloading policy between multiple fog nodes in a ring topology. In [21], a distributed policy for tasks assignment that can be executed efficiently in the network edge cloud has been proposed. The author has not considered the communication between fog-to-cloud and IoT-to-cloud. This model's scalability is limited since the cloud servers send their status continuously to the mobile subscribers. It will not be comfortable with an immense amount of edge devices.

The authors in [22] proposed a new load balancing technique for fog nodes by combing graph partitioning theory and fog computing characterizing. To achieve a dynamic load balancing in fog computing, the authors considered graph repartitioning.

For managing a massive amount of data in a cloud environment with low cost, the authors in [23] replaced physical network balancers with virtualized network balancers. The virtualized network balancer consists of two parts; the first load is a master, and the other acts as a secondary, which includes network load balancers and load balancer selector.

This kind of balancer is better than a hardware balancer since the cost is reduced and the user can efficiently add or remove an algorithm to the system. The authors in [24] proposed a cooperative load-balancing model for fog/edge data centres to mitigate the delay services. The idea is to assign a specific buffer for each data centre to receive requests from other nodes. Once the number of requests exceeds a certain threshold, the coming request is moved or balanced to an adjacent node. This kind of work anticipates the nodes are connected by the high-speed connection for achieving effective load balancing.

Based on the literature review and to the best of our knowledge, there is no work yet that employs mobile agents, resource capabilities, and considering idle fog nodes to build a

fog-based resource management model for enhancing the performance of big data application in IoT environment and improving fog computing resource utilization.

A new formulation is introduced for combined Cloud-Fog architectures [25]. The formulation reduced the service latency with the fulfilment of the Quality of Service (QoS) requirements. Moreover, the author used Gurobi Optimizer for addressing the Integer Linear Programming (ILP) model. In [26], the authors focus on the application models that increase the application deployment region. Also, they considered the placement strategy on edge and cloud platforms. The author presented a framework that increases the utilization of fog resources [27]. When a service is requested, the provisioning plan is implemented. Considering the workload is mentioned in [28], a new policy is proposed to determine the workload allocation on Fog-Cloud computing services considering the trade-off between the delay and power consumption. The authors split the original problem into three sub-problems in order to address each sub-problem separately. Three methods have been used in this framework; Generalized Benders Decomposition, convex optimization, and Hungarian. The authors in [29] provided a new model that is based on the mathematical service placement for the fog computing environment. This research aims to reduce the blocking probability, the percentage between the rejected workloads and the total workloads. The purpose of the research in [30] is to reduce network usage by presenting an optimization policy for data placement in the fog environment. This can be achieved by finding out the closest path between the fog device and the data source (IoT device). Minimizing the response time and maximizing the throughput are achieved in [31]. The algorithm distributes the workload on the fog resources environment. A job scheduling technique is also applied for Virtual Machines (VM) based on the service level agreement. In [32], the authors proposed a system to allocate and offload the service between the cloud server and fog computing. The decision rule relies on three conditions: completion time, services sizes, and the capacity of fog resources. Anther algorithm is proposed to satisfy the Service Level Agreement (SLA) and Quality of Service (QoS) and enhance the major data distribution in fog and cloud environments. Finally, the services mapping based on their priority level, the highest one would be mapped first, and so on. A new service placement framework is proposed in [33]. The authors attempt to reduce the latency considering the cost budget constraints. The Lyapunov optimization function is used in this framework to split the main problem into a set of problems with not considering user mobility. The author in [34] used machine learning to minimize the service costs and maintain the QoE. The Q-learning has been applied for defining the optimal migration for each service request. The authors in [35] demonstrate some of the service placement strategies in Edge-Cloud computing environments. This research aims to minimize the failed requests by formulating the problem as Mixed Integer Linear Programming (MILP). Two scheduling policies are used in this research: Earliest Deadline First (EDF), and First-In-First-Out (FIFO). The problem of dynamically deploying applications on fog resources, which should satisfy the Quality of Service (QoS) constraints, has been discussed in [36]. The authors expressed the previous problem as Integer Non-Linear Programming (INLP). Two heuristics are used to address the problem: a) Min-Cost: it is used to reduce the overall cost. b) Min-Vol: it is used for reducing deadline violations. The authors in [37] proposed a methodology to illustrate when and where the services should be placed. The placement strategy is based on the request ratio and user mobility in the edge network. The issue is modelled as a sequential decision-making Markov Decision Problem (MDP). Then the authors apply Lyapunov optimization on the two divided MDPs. As a result, the cost is reduced for each of the location constraints, delay, and execution. In [38], the research reflects the data locality. The author's design architecture consists of three tiers. The aims are to dynamically route the data to an optimal server and optimize the computing capacity. The prototype was implemented on the OpenStack virtualization environment by integrating the Software-Defined Network and Network Functions Virtualization (NFV). The architecture implemented on IoT surveillance system application, also a specific scheme is proposed in case of an urgent situation. Considering the load balancing to reduce the fog nodes' power consumption only is proposed in [39]. The author proposed an algorithm to allocate the fog resource efficiently. This algorithm is based on ordering the fog resources increasingly according to two factors: the availability and capacity to serve more tasks. Then assigning a threshold for each resource to keep them in a stack this mechanism helps utilize all available resources in the network. The result shows that the power consumption is reduced slightly compared to load balancing algorithms such as Round Robin and Throttled.

The load balancing and task distribution policy play a significant factor in optimizing the fog system's application performance. The centralized load balancing controller must gather information about all network devices to generate global optimization decisions. However, this kind of controller may not be efficient on some applications since all the devices should send the applications to a manager. The centralized core will generate the decision. Besides, one of the centralized controller dilemmas is a single point of failure that makes the system weak. On the other side, the decentralized load balancing should not gather all the information of all devices in the network, so many managers are connected in this kind of controller. Also, make a decision is not on a single core as in the centralized controller, which makes the scalability in the decentralized controller is higher than a centralized one.

Moreover, a decentralized controller's performance exceeds a centralized one since network overhead is high in the centralized controller. Overall, it is better to adopt the centralized and decentralized approaches in a new approach that can overcome both approaches' limitations.

The task distribution or job scheduling approaches can be divided into static and dynamic. The necessary information about the demands and available resources has been accomplished in the static approach before receiving the tasks. Also, the tasks would be sent at one time, and the scheduling decision has already been made. This approach is not suitable for the fog system because it is not easy to have all the necessary information about all devices in fog networks before the execution time.

However, in the dynamic approach, the scheduling process is made once the task is received in the system. It is also efficient to build a hybrid approach that makes the fog system works more effectively with different demands and applications. A summary of some previous works, based on the load balancing controller and task distribution policy, is provided in Table I.

TABLE I. COMPARISON OF SOME WORKS BASED ON THE CONTROLLER AND POLICY

Ref. No.	Load Balancing Controller		Task Distribution Policy	
	Centralized	Distributed	Static	Dynamic
[34]	✓			✓
[33]	✓			✓
[25]		✓	✓	
[26]		✓	✓	
[32]		✓	✓	
[28]	✓		✓	
[29]	✓		✓	
[30]	✓		✓	
[35]	✓			✓
[36]	✓			✓
[37]	✓			✓
[38]	✓			✓
[31]		✓	✓	
[32]		✓	✓	

III. PROPOSED MODEL

The proposed model aims to mitigate the drawbacks mentioned in the previous section. Initially, there is a need for a new method to handle the increase of incoming tasks from IoT devices. This can be handled by building an efficient job scheduling technique and effective job placement mechanism. Secondly, since IoT devices have been increasing recently, numerous data need to be proceed and analysed. The mobile agent software is involved in this model to reduce network cost by transferring the necessary data from the cloud server.

In our proposed model donates to the tasks that are sent from the IoT devices. The task priority plays a significant factor in reducing the responding time. On the other hand, an efficient resource management system will mitigate the cost of determining the suitable fog node from enormous resources, executing the tasks, reducing the delay, and saving power consumption by utilizing all available resources. This model consists of three main components: job scheduling, job placement, and mobile agent software. The job scheduling has a primary duty to determine the task type: mobile agent or not. Also sorts the tasks depending on the priority that is assigned from the application requirement. The mobile agent is responsible for dealing with tasks requiring service on the cloud server, such as inquiries in the cloud data center. The job placement sorts and ranks the available resources increasingly by free space for jobs and tracking each resource's status.

In Fig. 2, the IoT devices send a set of tasks $T = \{t_1,\ldots,t_m\}$ continuously to the fog layer. The FNAMM model receives the sent tasks to be executed in the fog nodes or cloud side. Initially, the model scans the network for discovering a set of resources $N = \{n_1,\ldots,n_m\}$ sort the incoming tasks by their accompanied priorities. Each fog area includes one master fog node (MFN) and many fog nodes (FNs) attached to the master fog node. The master node receives a series of tasks <M,P,R>, where M is the task type mobile agent or not. The mobile agent will be forwarded to the cloud server, and not the mobile agent will be executed in local fog resource. The P defines the task priority, and R indicates the fog resources' availability in that area. If the task cannot be executed in this area, the master node will migrate the task to execute in the neighbour fog area instead of sending it to the cloud server and so on. This will reduce the delay by implementing the task as locally as possible.

A. Optimized fog Topology Job Scheduling (OFTJS)

This section proposes an optimized fog topology job scheduling (OFTJS) algorithm proposed in our solution in [40]. Most fog computing systems use the FCFS algorithm, which executes one job at a time. This strategy is not efficient when the system is dealing with a massive number of jobs. Moreover, the job priority is not considered in this strategy as well.

Suppose the system topology consists of 6 main areas, and each area has 10 fog resource nodes. So, we have 60 fog resources that can execute the job in a fog computing network. When any nodes in the system cannot accept any more jobs, they would be migrated to the cloud side. In the proposed approach, we add a job pool between the incoming jobs and the system. The model's size is L, which is the number of jobs to be executed in the system, as shown in Fig. 1.

Fig. 1. Job Scheduling Process.

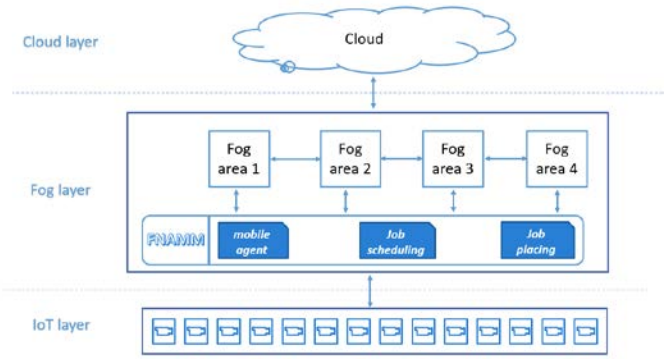

Fig. 2. High-Level Architecture for the Proposed Model.

Once the scheduling process starts, all the jobs would be placed into the job pool and allocated to the fog system's appropriate nodes. Also, the devices in the system would be scanned in each periodical scheduling cycle. The purpose of the scanning technique is to detect all available resources and their capabilities in the system. After determining the free and suitable resources in the system, we acquire a set of waiting jobs in the job pool ordered by the priority, as shown in Fig. 3.

The applications in IoT/fog computing environment have their requirements and characteristic. The end-user sends tasks to the fog layer to being executed then achieving the result. However, sorting the tasks in a queue is different from one application to another one. For instance, an eHealth application will give the tasks high priority if the patient has a high blood pressure to execute early. The priority mechanism is based on the task's type. In other words, each task has a deadline to be completed depending on the application requirements, as shown in Fig. 5. Based on the application requirement, we suggest a priority scheduling for the incoming tasks according to two factors:

1) The task would reach its threshold so that it will be considered a high priority.

2) The task has already been assigned as a high priority through the applanation requirement.

Algorithm1: optimized fog topology job scheduling (OFTJS)

1. **If** scheduling cycle s is launched **then**

2. scan the fog system and discover the set N of M free resources: N = {n_1,…,n_M}

3. gather the set t of T from job pool: T = {t_1,…,t_m}

4. **if** task_type(t_i) == mobile_agent **then**

initates_mobile_agent_toCLoud(t_i)

else

Job Placement (J , N) → Algorithm 2

7. **If** all the tasks in T are executed **then**

terminate the scheduling cycle s+1

else if t_i € T is rejected **then**

if service_not_aval(t_i) == true **then**

migrate_to_cloud(t_i)

terminate the scheduling cycle s+1

else

reserve space in job pool

Fig. 3. Job Scheduling Algorithm.

Algorithm2: job placement

Input: i) the set N of M nodes: N= { n_1, n_2, … , n_m }

ii) the set t of T waiting jobs in the task-pool: T = { t_1, t_2 , … , t_m}

1. sort and rank each n_i increasingly by free space for tasks

2. PR = priority_assign(t) → Algorithm 3

3. **if** PR_i == H **then**

place t_i in TP^H *// high task pool*

else

place t_i in TP^N *// normal task pool*

4. **for** each task € TP^H **DO** // *high task pool placement*

5. scan the system to obtain updated set N of free fog nodes

6. **if** n_i has more space for TP^H_i **then**

return placing TP^H_i in n_i

else

continue

7. **for** each task € TP^N **DO** // *normal task pool placement*

8. scan the system to obtain updated set N of free fog nodes

9. **if** n_i has more space for TP^N_i **then**

return placing TP^N_i in n_i

else

continue

Fig. 4. Job Placement Algorithm.

B. Job Placement

The job placement algorithm plays a significant role in reducing the power consumption and the response time by placing the task close to the IoT device. Moreover, selecting the task to be executed early is essential for achieving the (QoS) and (SLA). In our job placement algorithm, it receives the tasks and the available and suitable resources. The first step is to sort the resources increasingly by free space to execute a task. Secondly, the priority function assigns priority for each task, as explained in the previous paragraph. Thirdly, if the task is assigned as a high priority, it will be placed in the high task pool; otherwise, it will be placed in the normal task pool, as shown in Fig. 4.

Algorithm3: priority assignment

Input: the set t of T waiting jobs in the task-pool: T = { t_1, t_2 , … , t_m}

Step1: /* assign the task that will reach the threshold */

If $delay_i^t == TH_i$ **then**

return (H)

Step2: /* assign the task that has high priority given by the application requirements */

If $Tpr_i == 1$ **then**

return (H)

else

return (N)

Fig. 5. Task Priority Assignment Algorithm.

C. Mobile Agent Software Technology

Once the job scheduling, as mentioned in the previous section, determines the task as a mobile agent software, the task will be considered as a mobile IoT agent. When the mobile IoT agent is launched, the discovery manager requests the cloud service pool to provide a set of available virtual machines in the cloud layer, high speed, and high processing power devices. Moreover, it determines the required service from the caller/IoT. Finally, the discovery manager generates an action plan including routing decisions for the mobile IoT agent, as shown in Fig. 6 and Fig. 7.

Upon the fog layer's migration to the cloud layer, the execution and data transmission paths select the same bridge. If the connection between the fog and the cloud is interrupted, the mobile IoT agent may remain on the cloud side till the caller reconnects to achieve the result.

The model consists of three main components as follows:

- Discovery manager: this agent aims to provide the available Virtual Machines in the cloud server and calculates the bandwidth between the caller/IoT and the VM host; as a result, the execution time would be minimized. This method can be achieved by sending a request to the cloud service pool to provide the available VMs in the cloud server.
- Cloud service pool: the cloud service pool is a database that continuously provides VMs hosts that implement the mobile IoT agent. All VMs hosts' specifications are provided by this database, such as CPU speed and cores number, storage size, and current capacity.
- Cloud VMs: these machines are available in the cloud layer for executing the incoming mobile IoT agent's task. In this case, the service is provided as platform as a service (PaaS) from the cloud server.

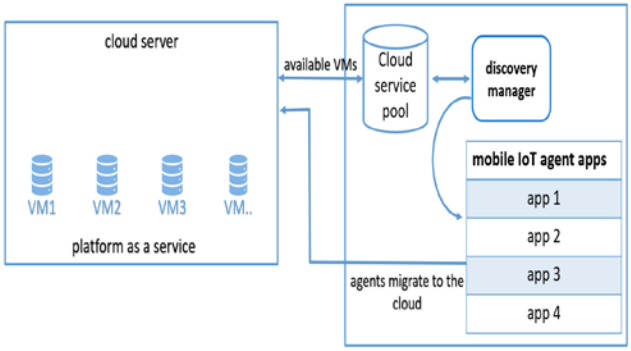

Fig. 6. Mobile Agent Architecture.

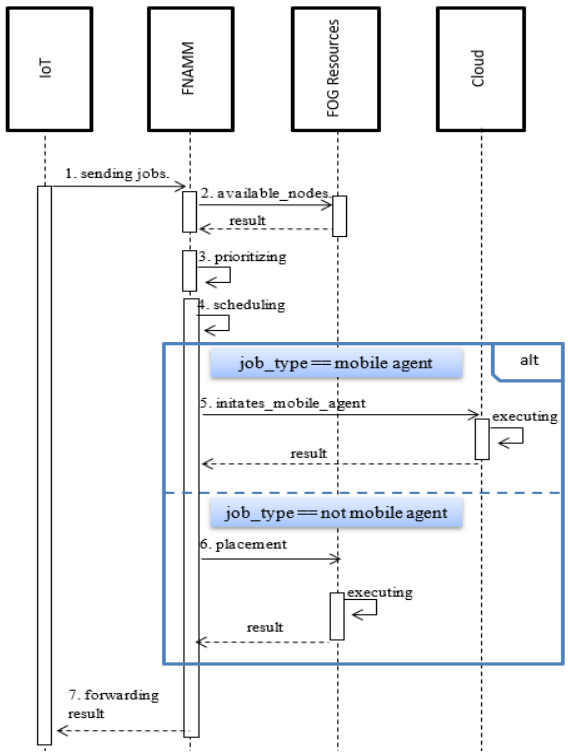

Fig. 7. FNAMM Sequence Diagram.

IV. Discussion

The proposed solution's effectiveness compared to [25] and [38] architectures deals with high efficiency when dealing with big data in the cloud. In the proposed solution, we used a mobile agent to reduce the volume of data that is transferred between the end-user and the cloud server, which also contributed to reducing the cost of the network as well. As for the fog network, our task scheduling tries as much as possible to implement tasks locally, near the end-user. On both [25] and [38], when the device cannot perform the tasks, it sends it north towards the cloud server. While in the proposed solution, we try to implement the tasks in devices that are adjacent to this device, taking into account both the left and right directions. Finally, the proposed model differs from the compared architectures in that the task priority collaborates in our solution. Each IoT applications have their requirements that can affect the task priority. So, depending on the

application requirements, the proposed task scheduling algorithm will regard these requirements in sorting the task in the queue.

Most of the recent architectures have not considered a massive amount of process in fog computing networks. When we compare it with other architectures, the proposed architecture's significant feature is considering the data velocity in the IoT environment. We can optimize the performance and scalability by building an efficient resource management model. Creating a repository that contains all available resources/service and their capability in the fog network can enhance task scheduling and load balancing. Also, the metadata in the repository can indicate the data locality and then decide if it would be implemented in the fog network or must be migrated to the cloud side in early stage.

The strength of this architecture can be demonstrated in the next point:

- Dynamic: the architecture supports the collaboration between the resources to scale the dynamic changes in the network. Also, the collaboration between the networks is dynamic, which can enhance the join process.
- Saving energy: since the architecture focuses on utilizing all the resources in the networks, the transferred process to the central cloud would be reduced.
- Response Time: the architecture determines the short path between the resource and the destination, leading to reduced latency, also, by early determining, on the distribution task phase, if the job would be executed in the fog resource or on the cloud server.

It is insufficient to use traditional methods when required data is transferred from the cloud servers to the user or IoT devices. In some cases, unused data is transmitted; thus, there is a waste of energy and delays in responding to demands. From this challenge, mobile agent technology which does analysis or processing on the cloud side, then transmits target data in a small amount to the end-user.

V. Conclusion and Future Work

IoT applications generate massive tasks that need to be served adequately and received a fast-responding. Fog computing is proposed to accommodate the cloud server by providing the service close to IoT devices. However, many fog computing architectures are insufficient to utilize all available resources. A holistic fog-based resource management model is proposed to overcome the mentioned issue by building an efficient job scheduling and deploy the job to appropriate available resources considering capabilities. Our proposed model's benefits can be summarised in making a reduction in response time, network cost, and power consumption. These metrics play a significant factor in optimizing the performance of IoT applications. The future work is to implement this model in a simulation work or a real-time environment. Moreover, the mobility of IoT devices is not considered in our solution, which can be investigated in further research.

References

[1] K. Kambatla, “Trends in big data analytics”, J. Parallel Distrib. Comput., vol. 74, no. 7, pp. 2561-2573, 2014.

[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, Sateesh Addepalli, “Fog computing and its role in the internet of things”, Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16, 2012.

[3] Mckinsy Global Institute website “The Internet of Things: Mapping The Value Beyond The Hype”. Accessed Jan. 10, 2021. [Online]. Available: https://www.mckinsey.com.

[4] A. V. Dastjerdi, H. Gupta, R. N. Calheiros, S. K. Ghosh, and R. Buyya, “Fog computing: Principles, architectures, and applications,” in Internet of Things, Elsevier, ch.4, pp. 61–75, 2016.

[5] J. Noymanee, W. San-Um, Theeramunkong, “T. A Conceptual Framework for the Design of an Urban Flood Early-Warning System Using a Context-Awareness Approach in Internet-of-Things Platform”, In: Kim K., Joukov N. (eds) Information Science and Applications (ICISA) 2016.

[6] P. Hu, H. Ning, T. Qiu, Y. Zhang, X. Luo, “Fog computing based face identification and resolution scheme in Internet of Things”, IEEE Trans. Ind. Informat., vol. 13, no. 4, pp. 1910-1920, Aug. 2017.

[7] I. Gudymenko, K. Borcea-Pfitzmann, & K. Tietze, “Privacy implications of the Internet of Things”, in International Joint Conference on Ambient Intelligence, Springer Berlin Heidelberg, pp. 280-286, 2011.

[8] M. Aazam, S. Zeadally, K. A. Harras, “Fog Computing Architecture Evaluation and Future Research Directions’’, in IEEE Communications Magazine, vol. 56, no. 5, pp. 46-52, 2018.

[9] O. Kwon and N. B. L. Shin, “Data quality management, data usage experience and acquisition intention of big data analytics”, Int. J. Inf. Manage., vol. 34, no. 3, pp. 387–394, 2014.

[10] M. Marjani et al., “Big IoT data analytics: Architecture opportunities and open research challenges”, IEEE Access, vol. 5, pp. 5247-5261, 2017.

[11] K. Peng, R. Lin, B. Huang, H. Zou, and F. Yang, “Link importance evaluation of data center network based on maximum flow,” Journal of Internet Technology, vol.18, no.1, pp.23–31, 2017.

[12] X. Xu, X. Zhang, M. Khan, W. Dou, S. Xue, S. Yu, “A balanced virtual machine scheduling method for energy-performance trade-offs in cyber-physical cloud systems”, Future Generation Computer Systems, 2017.

[13] B. Varghese, N. Wang, S. Barbhuiya, P. Kilpatrick, and D. S. Nikolopoulos, “Challenges and Opportunities in Edge Computing”, in IEEE International Conference on Smart Cloud, pp. 20–26, 2016.

[14] B. Varghese, N. Wang, J. Li, and D. Nikolopoulos, “Edge-as-a-Service: Towards Distributed Cloud Architectures”, in International Conference on Parallel Computing, ser. Advances in Parallel Computing. IOS Press, pp. 784–793, 2017.

[15] E. Saurez, K. Hong, D. Lillethun, U. Ramachandran, and B. Ottenwälder, “Incremental Deployment and Migration of Geo-distributed Situation Awareness Applications in the Fog”, in Proceedings of the 10th ACM International Conference on Distributed and Event-based Systems, pp. 258–269, 2016.

[16] R. Kolcun, D. Boyle, and J. A. McCann, “Optimal processing node discovery algorithm for distributed computing in IoT”, in 5th International Conference on the Internet of Things, pp. 72–79, 2015.

[17] Xu, X.; Fu, S.; Cai, Q.; Tian, W.; Liu, W.; Dou, W.; Sun, X.; Liu, A.X., “Dynamic Resource Allocation for Load Balancing in Fog Environment”, Wirel. Commun. Mob. Comput., 2018.

[18] W. Tärneberg, A. Mehta, E. Wadbro, J. Tordsson, J. Eker, M. Kihl, E. Elmroth, “Dynamic application placement in the mobile cloud network”, Future Generation ComputerSystems, vol. 70, pp. 163-177, 2017.

[19] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, G. Mastorakis, “Drop computing: Ad-hoc dynamic collaborative computing”, Future Gener. Comput. Syst., vol. 92, pp. 889-899, Mar. 2017.

[20] C. Fricker, F. Guillemin, P. Robert, and G. Thompson, “Analysis of an offloading scheme for data centers in the framework of fog computing”,

ACM Trans. Model. Perform. Eval. Comput. Syst., vol. 1, no. 4, p. 16, 2016.

[21] X. Guo, R. Singh, T. Zhao, Z. Niu, “An index based task assignment policy for achieving optimal power-delay tradeoff in edge cloud systems”, Proc. IEEE Int. Conf. Commun. (ICC), pp. 1-7, May 2016.

[22] S. Ningning, G. Chao, A. Xingshuo and Z. Qiang, “Fog computing dynamic load balancing mechanism based on graph repartitioning”, in China Communications, vol. 13, no. 3, pp. 156-164, Mar. 2016.

[23] Po-Huei Liang and Jiann-Min Yang, “Evaluation of two level global load balancing framework in Cloud Environment”, International Journal of Computer Science and Information Technology (IJCSIT), Vol. 7 No 2, Apr. 2015.

[24] R. Beraldi, A. Mtibaa, and H. Alnuweiri, “Cooperative Load Balancing Scheme for Edge Computing Resources”, in 2nd International Conference on Fog and Mobile Edge Computing. IEEE, pp. 94–100, 2017.

[25] V. B. C. Souza, W. Ramirez, X. Masip-Bruin, E. Marin-Tordera, G. Ren, and G. Tashakor, “Handling service allocation in combined fog-cloud scenarios,” Proc. IEEE Int. Conf. Commun. (ICC), Kuala Lumpur, Malaysia, pp. 1-5, 2016.

[26] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti. “Cutting Throughput on the Edge: App-Aware Placement in Fog Computing”. Proc. 6th IEEE Int. Conf. (CSCloud), pp. 196-203, 2019.

[27] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, ‘‘Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption,’’ IEEE Internet Things J., vol. 3, no. 6, pp. 1171–1181, Dec. 2016.

[28] O. Skarlat, S. Schulte, M. Borkowski, and P. Leitner, ‘‘Resource provisioning for IoT services in the fog,’’ in Proc. IEEE 9th Int. Conf. ServiceOriented Comput. Appl. (SOCA), pp. 32–39, Nov. 2016.

[29] K. Intharawijitr, K. Iida, and H. Koga, “Analysis of fog model considering computing and communication latency in 5G cellular networks,” in Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops (PerCom Workshops), Sydney, NSW, Australia, pp. 1–4, 2016.

[30] I. Lera, C. Guerrero, and C. Juiz, ‘‘Comparing centrality indices for network usage optimization of data placement policies in fog devices,’’ in Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC), vol. 1, no. 1, pp. 115–122, Apr. 2018.

[31] 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, 2016.

[32] A. A. Alsaffar, H. P. Pham, C.-S. Hong, E.-N. Huh, M. Aazam, "An architecture of IoT service delegation and resource allocation based on collaboration between fog and cloud computing", Mobile Inf. Syst., vol. 2016, Aug. 2016.

[33] T. Ouyang, Z. Zhou, and X. Chen, ‘‘Follow me at the edge: Mobility-aware dynamic service placement for mobile edge computing,’’ IEEE J. Sel. Areas Commun., vol. 36, no. 10, pp. 2333–2345, Oct. 2018.

[34] M. Chen, W. Li, G. Fortino, Y. Hao, L. Hu, and I. Humar. (Aug. 2018). ‘‘A dynamic service-migration mechanism in edge cognitive computing.’’ [Online]. Available: https://arxiv.org/abs/1808.07198.

[35] A. Ascigil et al., "On uncoordinated service placement in edge-clouds", Proc. IEEE CloudCom, pp. 41-48, 2017.

[36] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, X. Wang, H. C. Cankaya, Q. Zhang, W. Xie, J. P. Jue, "FOGPLAN: A lightweight QoS–aware dynamic fog service provisioning framework", IEEE Internet Things J., vol. 6, pp. 5080-5096, Jun. 2019.

[37] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds", Perform. Eval., vol. 91, pp. 205-228, Sep. 2015.

[38] J. Wang, J. Pan, and F. Esposito, “Elastic urban video surveillance system using edge computing,” in Proceedings of the Workshop on Smart Internet of Things, ser. SmartIoT ’17. New York, NY, USA, pp. 7:1–7:6, 2017.

[39] B. H. Malik, M. N. Ali, S. Yousaf, M. Mehmood, H. Saleem, ‘‘Efficient Energy Utilization in Cloud Fog Environment, ’’ International Journal of Advanced Computer Science and Applications (IJACSA), Vol. 10, No. 4, 2019.

[40] A. Alghamdi, A. Alzahrani, V. Thayananthan, ‘‘Execution Time And Power Consumption Optimization In Fog Computing Environment,’’ International Journal of Computer Science and Network Security (IJCSNS), Vol. 21 No. 1 pp. 137-142, 2021.