Development of IoT Modules with Latency Considerations in Mixed Fog-Cloud Computing Environments

K.Mounika[1], T.Ravi Charan[2] , J.Pujitha[3]

[1] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

[2] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

[3] Assistant Professor, Department of CSE, Sri Indu Institute of Engineering & Technology, Hyderabad

Abstract: The modern paradigm of the Internet of Things (IoT) has led to a significant increase in demand for latency-sensitive applications in Fog-based cloud computing. However, such applications cannot meet strict quality of service (QoS) requirements. The large-scale deployment of IoT requires moreeffective use of network infrastructure to ensure QoS when processing big data. Generally, cloud-centric IoT application deployment involves different modules running on terminal devices and cloud servers. Fog devices with different computing capabilities must process the data generated by the end device, so deploying latency-sensitive applications in a heterogeneous fog com-puting environment is a difficult task. In addition, when there is an inconsistent connection delay between the fog and the terminal device, the deployment of such applications becomes more complicated. In this article, we propose an algorithm that can effectively place application modules on network nodes while considering connection delay, processing power, and sensing data vol- ume. Compared with traditional cloud computing deployment, we conducted simulations in iFogSim to confirm the effectiveness of the algorithm. The simulation results verify the effectiveness of the proposed algorithm in terms of end-to-end delay and network consumption. Therein, latency and execution time is insensitive to the number of sensors.

Keywords: IoT; fog-cloud computing architecture; module placement;latency sensitive; application; resource aware placement

1 Introduction

The Internet of Things (IoT) makes it possible to connect everyday devices to the Inter-net. The large amount of data generated by these connected objects causes computational and network load on the system and requires a large amount of storage space. In the near future, we will have billions of interconnected IoT devices that will generate large amounts of data for processing and storage. For example, the author estimated in [1] that there will be approximately

1 trillion IoT devices by 2025. According to this approximation, the potential economic impact of the IoT industry will be almost equal to 11% of the world economy [2]. Currently, most IoT applications are deployed using cloud servers, because cloud servers provide centralized large-scale processing and storage resources to process massive amounts of data generated by large-scale IoT applications. For the realization of IoT applications, the most reasonable architecture available is the cloud computing paradigm [3]. IoT applications generate a large amount of diverse data. However, because the end-to-end latency increases in proportion to the amount of data to be processed on the cloud server, the cloud architecture limits the large-scale implementation of IoT applications [4]. Therefore, when designing large scale IoT applications on the cloud paradigm, it will cause high network consumption and delays. Many applications require real-time processing of sensing data with strict latency requirements, so the traditional cloud computing paradigm cannot meet these strict Quality-of-Service (QoS) requirements of these IoT applications.

Various application program designs require application modules to be placed on multiple heterogeneous nodes with different processing and storage functions. Cloud computing provides high storage and processing capabilities for the centralized implementation of IoT applications. The cloud paradigm has many advantages, such as mobility, reliability, ease of deployment, cost-effectiveness, and so on. Compared with the traditional cloud architecture, the distribution and mobility of sensor devices in future applications require the distribution of resources in a distributed manner throughout the network. On the other hand, Fog-cloud computing is emerging as a new paradigm to be used with cloud computing to deploy most of the upcoming IoT applications. The fog computing architecture can solve the above challenges through its additional advantages, such as dynamic scalability, reduced latency, and low consumption costs on cloud servers [5]. The data arrival rate towards these devices depends on the sensing interval of the edge node/IoT device. When the arrival rate of the input becomes higher, it will increase the overhead on the parent node. However, fog devices are energy-constrained devices, and their computing resources are limited, so they cannot bear such a large overhead for a long enough time [6].

The fog and edge computing paradigm are of much interest in the design and development of IoT applications. The fog paradigm transfers computing power at the edge of the system, and needs to be modified during the design process for the design and implementation of IoT applications. Applications that are sensitive to latency require processing capabilities near the edge of the network. Due to the centralization of resource provision, this specific function is missing in the cloud computing architecture. Fog computing provides a new resource layer in the entire network between the cloud and edge devices in a distributed manner. By using these limited fog layer resources to preprocess the sensing data from IoT devices, the delay can be minimized. A strategy and algorithm need to be designed to ensure the best use of available resources to implement latency-sensitive applications on the fog cloud paradigm. In this article, we propose an algorithm that can effectively allocate application modules by effectively using the network, so as to achieve the lowest delay while effectively consuming network resources. The module placement method proposed in this paper attempts to maintain a balance between the load generated by the edge node and the resources available in the parent fog devices. In addition, the proposed algorithm allocates application modules in a way that achieves the minimum latency time of the application.

The main contributions of this research are summarized as follows:

. We propose a module placement algorithm for the fog cloud architecture, which can handle the dynamic configuration of different parameters. We deploy the application in a heterogeneous fog environment in a way to achieve effective use of network resources.

- We suggest allocation of application modules on the heterogeneous fog nodes according to their processing capacity and load from edge devices. We place application modules on selective nodes to implement mandatory delay requirements for delay-sensitive applications.
- The algorithm was compared with cloud-based systems of different scales to verify the efficiency of its implementation for latency-sensitive applications. The parameters to be considered during the evaluation are latency, network consumption and the execution cost of cloud computing.

The structure of this article is as follows. Section 2 summarizes the related works. Section 3 presents the system model and formulates the problem. Section 4 describes the proposed module placement algorithm. Section 5 discusses the simulation results. In the last section, we draw conclusions.

2 Related Works

This section introduces the previously designed related work and algorithms related to application placement and module allocation strategies in the fog cloud architecture.

In [7], the author proposed an IoT application placement algorithm that has a dualistic approach to improve overall system performance, but it needs to be weighed against a slight increase in power consumption. In this method, the priority considerations for IoT application placement requests such as environment runtime context, application usage and user expectations, and QoS violations are regarded as feedback. After prioritizing the application placement request, the modules are placed on the appropriate atomization device based on their computing resources, proximity and response time.

The deployment of IoT applications [8] and the use of fog computing paradigms in large-scale IoT applications [9] and context-aware applications [10] are still in their infancy. In [11], we can find a model for optimizing the energy consumption of fog computing. In [12], the author proposed a strategy to allocate workload to energy-constrained fog devices. Literature [13] proposed an optimized service allocation strategy in fog cloud deployment to improve performance. In [14], a QoS-conscious placement strategy was proposed, which placed delay-sensitive services on devices located at the edge of the network to reduce costs and delays in the fog cloud network. In [15], the author proposed a strategy to manage delay-sensitive applications through module placement and resource optimization without violating the QoS of application. In this article, the author proposes two algorithms. The first algorithm performs application module placement to meet the latency requirements of the application, and the second algorithm performs resource optimization to achieve the QoS of the application.

We can find the concept of fog colony first proposed in [16]. This concept has been copied in many related works. The fog colony consists of multiple fog cells and acts as a micro data center. A fog cell is an application module placed on a physical device to provide services and resources for IoT devices connected to it. In [17], the authors proposed a distributed application placement strategy for fog computing architecture based on contextual information. The contextual information used by their proposed algorithm is based on the location of the fog device, network status, service type, and QoS deadline for each application to provide resources for IoT applications. The authors use the IBM CPLEX solver to solve their optimization problems. The experimental results reduce the delay and maximize the efficiency of the fog network. However, when the number of fog nodes increases under the solution proposed in [17], the calculation time of the proposed solution increases exponentially.

In [18], the authors use sEMG sensors to monitor patients in a persistent vegetative state from a remote location. In [19], a system model to reduce the delay in medical applications based on the Internet of Things is proposed. Similarly, in [4], a comparison of implementations of a cloud and fog-based remote pain monitoring system is proposed, which uses sEMG signals for pain detection. All these systems are compared between the different parameters of deploying multi-purpose applications on the fog and cloud paradigm. In Tab. 1 we briefly compare the qualitative characteristics of our proposed system with existing systems. Literature [20] proposeda module placement strategy of the fog-cloud computing paradigm, which effectively utilizes network resources and places modules according to the available capacity of network nodes. This algorithm searches for eligible network nodes to meet the module placement requirements, and places modules on eligible fog nodes unless the fog layer is exhausted. Tab. 2 shows the com- parison between the algorithm and previous module placement schemes in terms of observation parameters and application service placement.

Table 1: Qualitative comparison of our proposed system with the existing systems

Reference	Paradigm	Latency	Execution cost	Network usage	Sensing interval	Sensor/ Actuator latencies
[1]	Cloud	Moderate	High	High	Static	Static
[3]	Fog-cloud	High	Medium	Low	Static	Static
[4]	Fog-cloud	Low	Medium	Low	Static	Static
[20]	Fog-cloud	Medium	Medium	Medium	Static	Static
*Proposed	Fog-cloud	Low	Low	Low	Variable	Variable

Table 2: Comparison of the proposed algorithm with the existing module placement schemes

Reference	Observed parameters during application module placement			
	Latency	Sensing rate	Capacity	Deadline
[14]	✗	✗	C	C
[16]	✗	✗	C	C
[20]	✗	✗	C	✗
*Proposed	C	C	C	✗

In [21], the authors proposed a heuristic algorithm that supports heterogeneous nodes in a distributed manner and provides effective consumption of network resources in a manner that achieves the minimum delay of application diversity. In [22], the authors proposed an extension of the fog computing architecture based on the Hungarian algorithm, in which the role of the fog layer is to manage resource supply issues to obtain minimal delay and sustainable services. The experimental results verify the reduction of coupling problems and the effective use of resources. In [23], the authors proposed a resource allocation strategy for the multi-region fog layer architecture to ensure low energy consumption and reduce latency. Simulation results confirmed with the existing solution, the delay was reduced by 16.81%, and the energy usage was reduced

by 17.75%. An algorithm for placing heterogeneous modules on different fog devices is proposed in [24], which improves network consumption and reduces total execution time. In [25], the authors proposed two cascading algorithms for application module segmentation based on throughput and application allocation on area-based organizations. The result of the experiment is to confirm that the performance of the proposed solution is improved compared to other solutions available in this situation. In [26], the authors proposed a location-aware framework which records the locations of fog devices exist in the network. Compare the proposed algorithm with the existing framework and observe the reduction in latency, RAM and CPU consumption.

3 System Model and Problem Formulation

Applications designed based on the integrated fog and cloud paradigm gain the advantages of both architectures, and as a centralized unlimited processing and storage service through the cloud, it provides low-latency and large-scale distributed resource allocation. This integration of the two computing paradigms achieves many benefits, such as mobility, heterogeneity, and interoperability.

Fig. 1 shows the general integration of fog computing functions in the cloud architecture to provide distributed computing closer to the edge of the network. The three layers shown in the figure consist of devices with different resources and configurations. The first layer of the architecture consists of IoT devices that connect sensors and actuators. The fog layer contains fog devices that provide limited computing and storage resources in a distributed manner for preliminary processing. The cloud server on the third layer provides a large amount of storage and computing resources for processing and storing the data of all atomization devices connected to it. Each layer deals with specific components of the application, which will be further defined in the next section.

Each device in the infrastructure can accommodate application modules [27]. Among the various types of devices available in the network, our main concern is the smart gateway device located on the second layer, which links the first and third layers and provides resources for the basic processing of tasks. Each device belongs to a specific layer with different computing capabilities, and the specific layer contributes and defines the overall resource capacity of the layer. Multiple fog devices with predefined limited computing and storage capabilities constitute a fog layer. The cloud server at the other end is a resource-rich device, and its resource capacity is larger than any device available in the entire network.

In a heterogeneous IoT environment, all the types of devices related to resources are different. The devices available in the network consist of fog nodes and IoT devices with different computing capabilities. The latency time of the device to its parent node depends on the distance between the two communicating nodes. We can characterize the constraints that limit the fog node based on resource capacity and communication delay. The general function of the fog device is limited by the above constraints, which include basic parameters, namely CPU, RAM, storage and delay. The proposed algorithm is flexible enough to include more parameters when required by the application.

Let f_i denotes the i_{th} fog device in the network, and the capacity of this node is expressed as:

$$C(f_i) = < CPU_i, RAM_i, Bandwidth_i > \quad (1)$$

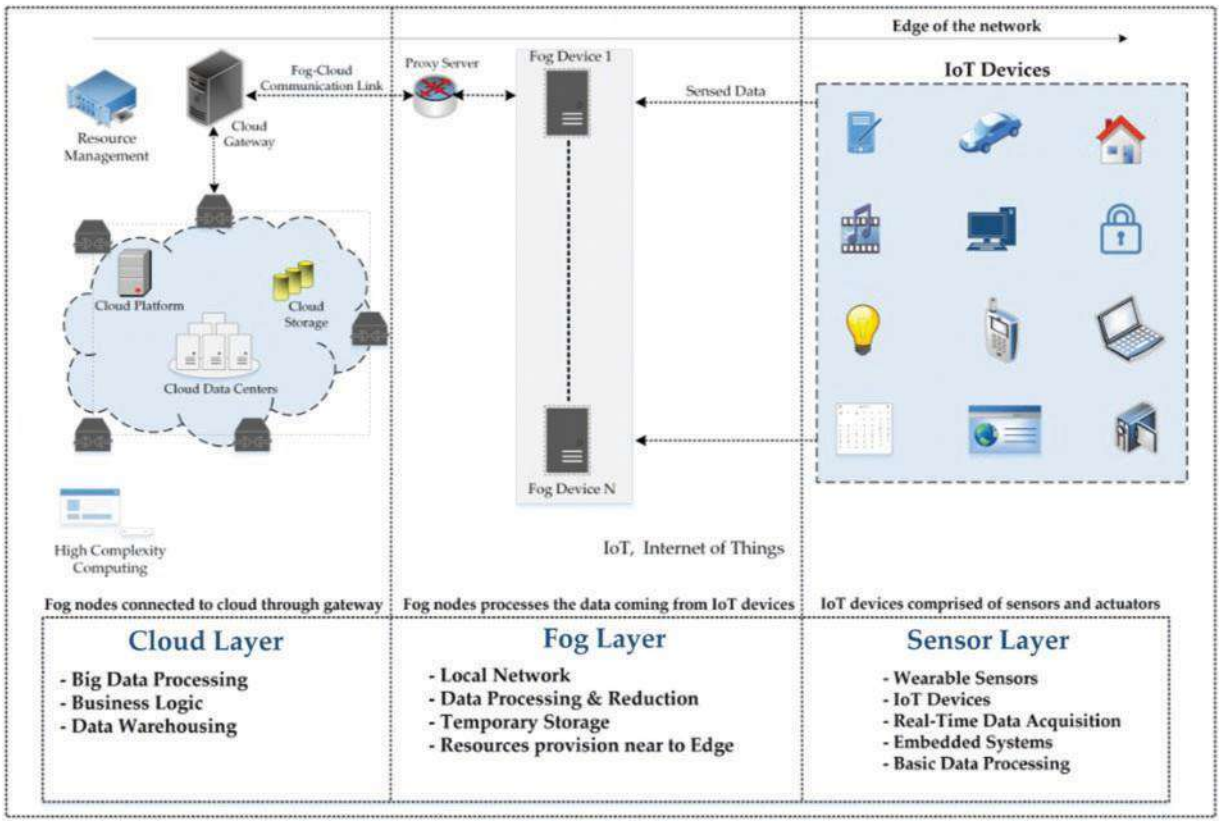

Figure 1: Fog-cloud architecture

The set of computing resources that define the entire network is the sum of the resources of all available devices in the network, expressed as,

$$N = \{f_i\} \tag{2}$$

Each device in the network passes some communication delay to its parent device, which depends on the distance between the communication devices, called latency. Let L_i^t is latency of the i_{th} node to its parent node where t is the tier in which the i_{th} device is located. The design method used in the application deployment based on the fog paradigm in this paper is the Distributed Data Flow Model (DDF) [6]. The DDF method can better understand the various components of the application and is the best way to deploy the application in a distributed computing environment.

As shown in Fig. 2, the application developed for simulating and evaluating the algorithm has three application modules as task processing elements and the interconnection between these application modules (used to deploy the application) is modeled as Directed Acyclic graph (DAG). In the DAG model, the vertex represents the task processing element of the application that processes the incoming data from the previous module. Arcs represent data dependencies between modules, and the flow of data from any module to another is also defined by arcs. The mathematical representation of our DAG model D of the application consists of vertices (V) and arcs (A), defined as,

$$D = \natural V, A) \tag{3}$$

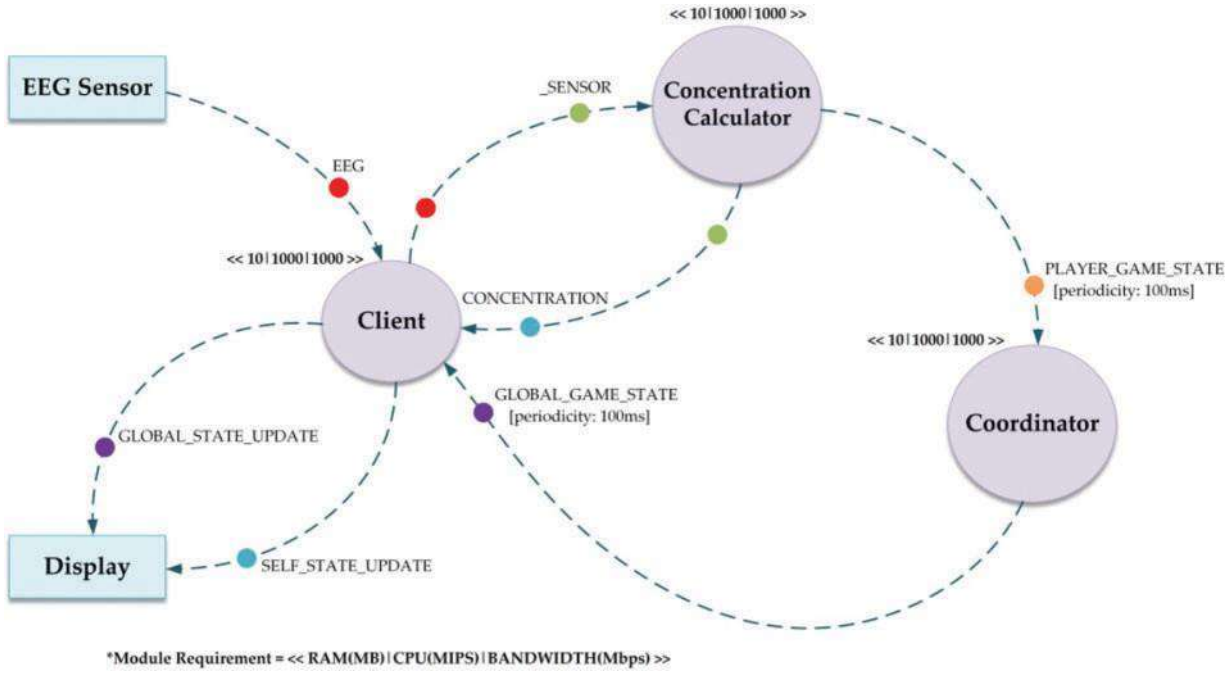

Figure 2: Directed acyclic graph of latency sensitive game deployed and modules are depicted using color-coding pairing in the graph

Each application module requires a certain amount of resources (i.e., CPU, RAM, bandwidth, and storage) to process tasks. Therefore, if m_i represents the i_{th} module of an application, then the requirement of that module is expressed as:

$$\text{Req}(m_i) = < CPU_i, RAM_i, Bandwidth_i > \tag{4}$$

The set M shows all the modules of the application.

$$M = \{m_i\} \tag{5}$$

The edges between application modules represent the data flow in the application. If e_i represents the edge originating from one module $m_i$to another, then the set containing all the edges of the application is denoted by E as in the following,

$$E = \{ e_i | e_i = < m_i, m_j > \} \quad \forall m_i, m_j \in M \tag{6}$$

The arcs connecting the vertices in the DAG model are periodic and event-based. The tuples generated by periodic edges are regularly performed after a specified time interval and event-based edges generate tuples if the source module receives a specific type of tuple according to the defined selectivity model.

We can calculate the latency offered by i_{th} fog device at any time t in the proposed Fog-Cloud architecture as follows,

$$L_t^i = L_{Sensor}^t + L_{edge}^t + L_{fog}^t \tag{7}$$

where, L^{t}_{Sensor} is the time consumed by the sensors in sensing EEG signals and L^{t}_{edge} is the delay caused by the end devices in transmitting signals to the fog nodes and L^{t}_{fog} is the latency offered by the i_{th} fog device to transfer the processed data to the parent node.

The module mapping function F indicates the placement of application modules on a network device that meets the requirement as defined in the following,

$$F : V \rightarrow N \tag{8}$$

$$\forall (m_i, f_i) \in F \mid \begin{array}{c} \Rightarrow \\ \forall m_i \in M \\ \forall f_i \in N \end{array} Req(m_i) < C(f_i) \mid \begin{array}{c} \Rightarrow \\ \forall m_i \in M \\ \forall f_i \in N \end{array} \Phi_{f_i}^{childs} \leq \Phi_{f_i}^{Threshold} \mid \begin{array}{c} \Rightarrow \\ \forall m_i \in M \\ \forall f_i \in N \end{array} L_t^{f_i} \leq L_{App}^{m_i} \tag{9}$$

where, $\Phi_{f_i}^{childs}$ and $\Phi_{f_i}^{Threshold}$ represents the sensing rate of the sensors deployed on the end devices connected to the i_{th} fog device and the threshold value of the i_{th} fog device to handle sensed data to achieve optimum performance, respectively. In the above equation L_{App}^{mi} is the threshold value of the latency defined by the application for the deployment of the application module to maintain latency requirements of the application. In the traditional process of deploying applications on the cloud paradigm, all modules are used on the cloud server, which leads to the high cost of cloud computing, and brings more delay and network consumption. Moreover, in the homogeneous fog paradigm that includes fog devices with equal resources, modules are statically distributed on the cloud and fog devices to provide services closer to edge devices. On the other hand, the heterogeneous fog network includes fog devices with different resources and different load conditions. In the heterogeneous fog paradigm, the application module is placed on a fog device that meets the requirements of the module.

We observed that the simulations are performed in the iFogSim toolbox developed for modeling and simulation of cloud and fog networks. This paper designs and redefines the EEG tractor beam game [28] to prove the effectiveness of this method. The application is based on the sensing process actuation model, where the sensed data is transmitted to the parent device in the form of a data stream for further processing by the application program module, and the generated instructions are sent to the actuator.

The DAG of the application model implemented in our simulation is shown in In Fig. 2, which contains five application modules including EEG, client, concentration calculator, coordinator and display. EEG stands for an IoT device that can sense the EEG signal of the player and send EEG-type tuples to the client module. In order to visualize the current state of the game to the user, the display module is placed on the actuator. In the computing environment, a tuple is the basic unit of communication between modules, and its characteristic is the specific length of data and processing requirements encapsulated in it. Tuples are defined by specific types, source modules and target modules, as shown by the DAG edge. The colored circles on the arc in the DAG represent the tuple mapping. For example, passing in a tuple of type _SENSOR on the module “Concentration Calculator” will result in an emission of tuple type CONCENTRATION. The client module is placed on the end device connected to the sensor and the actuator. The processing of all signals from all players requires a lot of processing and storage resources, so the coordinator module will be placed on the cloud server.

4 Proposed Solution

We propose two integrated algorithms here to place application modules in a heterogeneous Fog-Cloud environment to ensure reduced network consumption and latency to edge nodes. Our proposed algorithm will carefully check the fog nodes in the entire network and place application modules based on the available capacity, the sensitivity of the sensor, and the overall delay to the parent node.

The first algorithm is a module placement algorithm that takes fog nodes and application modules to place as input and returns the efficient placement of modules on suitable nodes. Set of heterogeneous fog devices F are first sorted in ascending order according to available capacity per node. After that, each fog node is examined to check its eligibility to handle the module to be placed. For every eligible node LOADCOMP algorithm is called that calculates the number of edge devices connected to that specific node and calculates the load on that fog device.

Algorithm 1 is a module placement algorithm, which places fog nodes and application modules as inputs, and returns the effective placement of the modules on the appropriate nodes. First, sort a group of different atomization devices F in ascending order according to the available capacity of each node. After that, each fog node will be checked to see if it is suitable for handling the modules to be placed. For each qualified node, Algorithm 2 (LOADCOMP) counts the number of edge devices connected to that specific node and calculates the load on the atomization device. A check is performed to confirm whether the ability of the fogging device is sufficient to handle the load provided by the edge device. The module is placed on the fog node to ensure the minimum latency threshold defined for the module. If the device is not suitable for placing the corresponding module, then the module is placed on the parent device. Fig. 3 shows the procedures involved in placing delay-sensitive modules on network nodes in the form of a flowchart.

Algorithm 1: Module Mapping Algorithm: Fog-Cloud Placement

Input: Set of Fog devices F and Application modules *M*
Output: Placement of application modules on network nodes

```
1:  function MODULEPLACEMENT (Fog Devices f_i ∈ Layer 2, Application Module to place m_i ∈ M)
2:  Sort (f_i); D inascendingorder
3:  placedmodules ← {};
4:  for each f_i do
5:    while (Cap (f_i) > Req (m_i)) do
6:      int i= LOADCOMP (f_i,m_i)
7:      if (i != −1) then
8:        Cap(f_i) = Cap(f_i) − Req(m_i);
9:      end
10:     else
11:       Place module on the Parent device
12:       add the module to placedmodules
13:     end
14:   end while
15:  end for
```

Algorithm 2: LOADCOMP algorithm for comparison of total processing load on Fog Device with available capacity

```
function LOADCOMP (Fog device x)
 Edges = {}
 for End device d ∈ Tier 3 do for all End devices
   if (Parent_d == x)
     Φ_SensingLoad = Φ_SensingLoad + Φ^d_SensingLoad
     Children_x = Children_x + 1
      add d to Edges
   end
 end for
  if ( Φ_SensingLoad / Children_x ≤ Φ^x_capability)
    for Child deviced ∈ Edges do for all Child devices
        if ((Φ^x_latency + Φ^d_latency + Φ^sensor_latency) ≤ (Latency^{x−module}_Threshold))
            Place module on device x
            add the module to placedmodules
          end
          else
            return -1;
          end
        end
      end
      else
        return -1;
      end
      end function
```

5 Results and Discussion

We simulated the EEG tractor beam game application to verify the effectiveness of the proposed algorithms. Our simulation environment has different numbers of sensor nodes with heterogeneous configurations in cloud and fog cloud paradigms. In our scheme, there are five fog nodes connected to the gateway node. Initially, each fog device was connected to two edge devices. We created the sensor in the simulation according to the strategy of [19]. In the simulation, we increased the number of sensors connected to each fog device to calculate latency and network usage. The graphical view of one of the scenarios created in iFogSim is shown in Fig. 4. In Tabs. 3–5, we have summarized the network parameters, tuple types and simulation settings used in our simulation scenarios.

The tuple emission rate of sensors rate is randomly selected between 10 and 30 milliseconds. We compared the proposed method with traditional cloud computing deployment in terms of latency, network usage, and cost of execution on the cloud. Several simulations were performed to equate the performance of the proposed fog-cloud method with the conventional cloud computing paradigm. Fig. 5 shows the network consumption of both paradigms in each simulation scenario. We observe large network consumption in the cloud-based implementation compared with the fog-cloud approach. In the proposed architecture, processing and storage functions are distributed throughout the system in the form of fog nodes. The network consumes less resource, because the fog node subsequently reduces the network load. On the contrary, in the cloud structure, all

the detected data is transmitted to the cloud server for further processing, resulting in a large consumption of network resources.

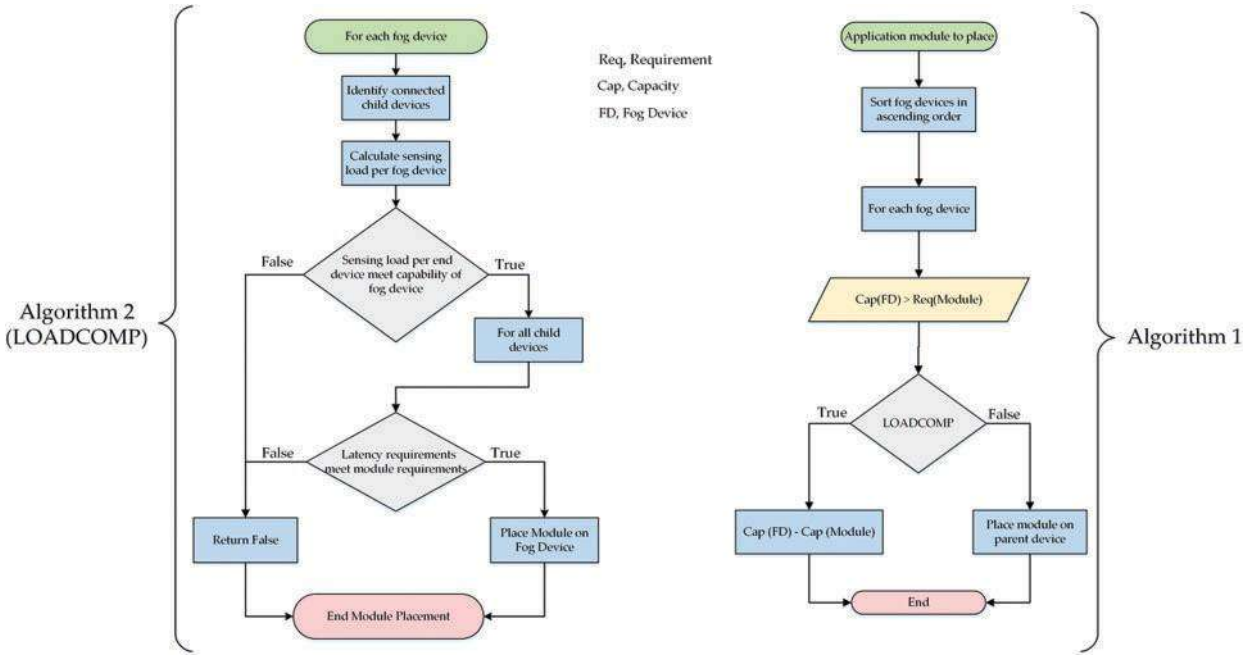

Figure 3: Flow diagram of proposed latency aware module placement algorithms

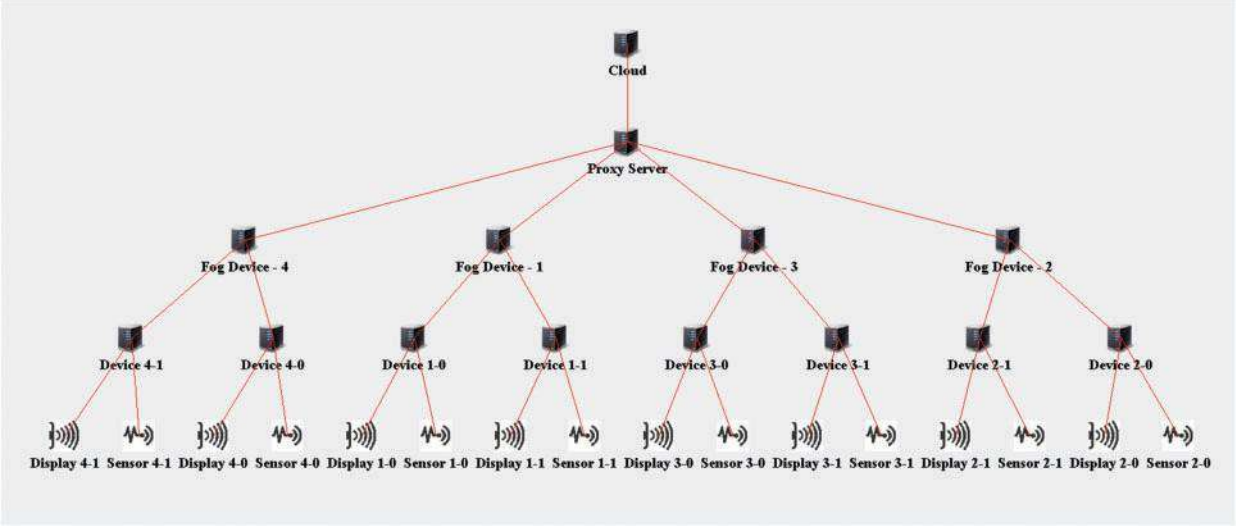

Figure 4: Initial topology in iFogSim for the evaluation of the proposed approach

Table 3: Network parameters used in simulations

Parameter	Cloud	Proxy	Fog node	End device
Level	0	1	2	3
Rate Per MIPS	0.01	0	0	0
RAM (GB)	40	4	4	4
Downstream bandwidth (MB)	10000	10000	10000	10000
CPU length (MIPS)	44800	2800	2800	2800
Upstream bandwidth (MB)	100	10000	10000	10000

Table 4: Tuple types and parameters used in simulation

Tuple type	Tuple CPU length (MIPS)	Network length
EEG	3000	500
_SENSOR	3500	500
PLAYER_GAME_STATE	1000	1000
CONCENTRATION	14	500
GLOBAL_GAME_STATE	1000	1000
SELF_STATE_UPDATE	1000	500
GLOBAL_STATE_UPDATE	1000	500

Table 5: Latency values used in simulations

Source	Designation	Latency (ms)
Cloud	Proxy server	100.0
Proxy server	Fog node	5.0
Fog node	End device	1.0–10.0
Device	Sensor	1.0–10.0
Device	Actuator	5.0–10.0

Fig. 6 shows a comparison between the latency provided by the cloud for each simulation scenario and the proposed fog cloud example. The latency provided by the cloud paradigm increases in proportion to the increase in the number of sensors connected to the cloud. In a cloud-centric architecture, the reason for the increase in latency is that the cloud server must process the data sensed by all sensors, while in the proposed fog-cloud method, the fog nodes process less data sensed by the sensors. The key feature of the proposed fog cloud algorithm is to provide fast response by using locally available resources at the fog node, thereby minimizing the processing load on the cloud server to the terminal device by executing tasks at the fog node close to the edge of the network and so reducing delay. The adaptability of the proposed algorithm in the fog cloud architecture reduces the amount of data to be executed to the cloud for processing,

which reduces the execution cost on the cloud, as shown in Fig. 7. This is due to the reduction in execution costs on the cloud, which provides additional processing power in the form of fog nodes between the cloud and edge devices.

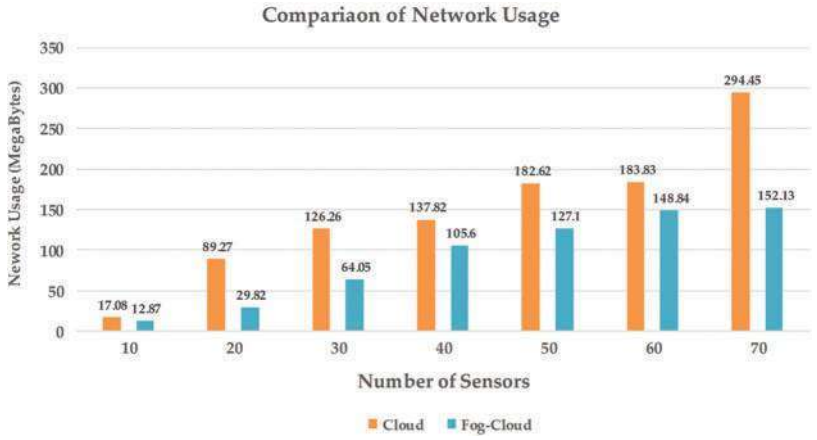

Figure 5: Network consumption in proposed fog-cloud *vs.* conventional cloud architecture

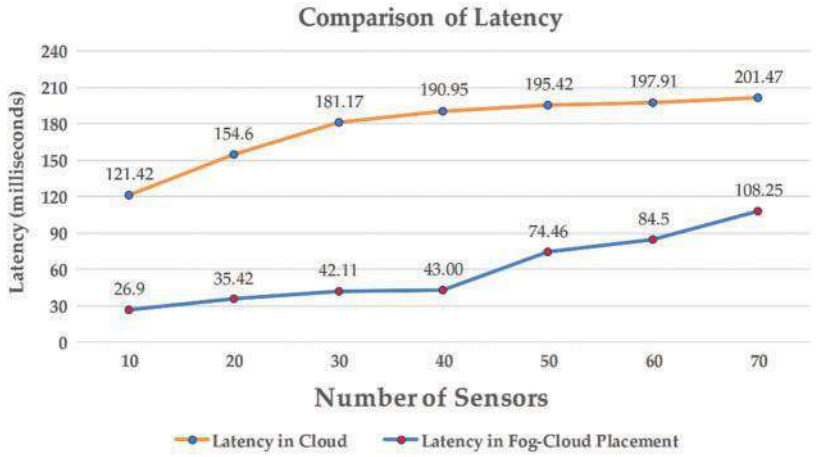

Figure 6: Latency in proposed fog-cloud *vs.* cloud computing architecture

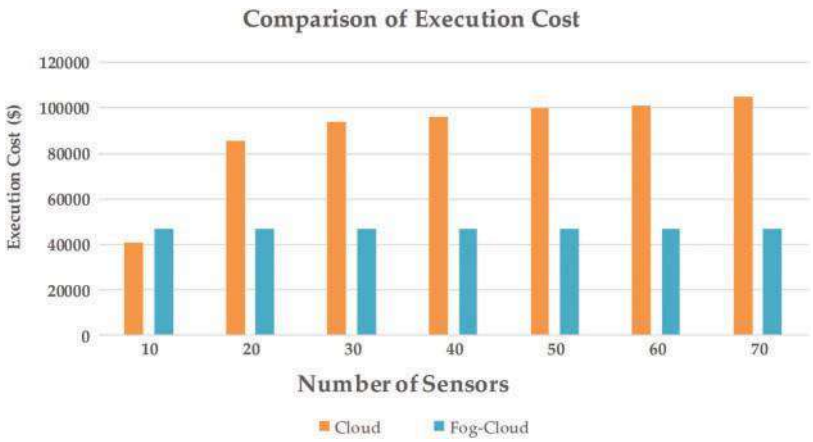

Figure 7: Execution cost in proposed fog-cloud *vs.* cloud-based implementation

6 Conclusions

In this article, we propose a module placement algorithm that can take into account the IoT applications and effectively place application modules in the Fog-Cloud computing paradigm. The proposed algorithm selects application modules and assigns them to appropriate devices to implement delay-sensitive IoT applications in a heterogeneous environment. In order to reduce the workload on the existing architecture, our proposed algorithm effectively places application modules on heterogeneous devices to reduce network usage. We used the proposed algorithm to develop a latency-sensitive game application to test the results. We have observed that compared with cloud architecture, using this algorithm under the fog cloud paradigm can significantly reduce the execution cost, end-to-end latency and network usage on the cloud. Our algorithm is equally applicable to all types of IoT applications. In the future, a new efficient scheduling algorithm for fog-cloud computing applications will attract widespread attention. We plan to implement more applications and further enhance algorithms for multiple features, and study the drawbacks and problems caused by node failures in the network.

Acknowledgement: This work was supported by the Deanship of Scientific Research at Prince Sattam bin Abdulaziz University, Saudi Arabia.

Funding Statement: This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2021-2016-0-00313) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. El-Hajj, A. Fadlallah, M. Chamoun and A. Serhrouchni, "A survey of internet of things (IoT) authentication schemes," *Sensors*, vol. 19, no. 5, pp. 1141–1184, 2019.

[2] R. Mahmud, K. Ramamohanarao and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Transactions on Internet Technology*, vol. 19, no. 1, pp. 1–21, 2018.

[3] C. S. Nandyala and H. K. Kim, "From cloud to fog and IoT-based real-time U-healthcare monitoring for smart homes and hospitals," *International Journal of Smart Home*, vol. 10, no. 2, pp. 187–196, 2016.
[4] S. R. Hassan, I. Ahmad, S. Ahmad, A. Alfaify and M. Shafiq, "Remote pain monitoring using fog computing for e-healthcare: An efficient architecture," *Sensors*, vol. 20, no. 22, pp. 6574–6595, 2020.
[5] R. M. Abdelmoneem, A. Benslimane and E. Shaaban, "Mobility-aware task scheduling in cloud-fog IoT-based healthcare architectures," *Computer Networks*, vol. 179, pp. 107348–107365, 2020.
[6] M. Ashouri, P. Davidsson and R. Spalazzese, "Quality attributes in edge computing for the internet of things: A systematic mapping study," *Internet of Things*, vol. 13, pp. 100346–100366, 2020.
[7] H. Nashaat, E. Ahmed and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.
[8] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.
[9] A. Brogi and S. Forti, "QoS-aware deployment of IoT applications through the fog," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1185–1192, 2017.
[10] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang *et al.*, "A survey on edge computing systems and tools," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1537–1562, 2019.
[11] M. A. Al Faruque and K. Vatanparvar, "Energy management-as-a-service over fog computing platform," *IEEE Internet of Things Journal*, vol. 3, no. 2, pp. 161–169, 2015.
[12] 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 of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.
[13] M. A. Ferrag, L. Shu, X. Yang, A. Derhab and L. Maglaras, "Security and privacy for green IoT-based agriculture: Review, blockchain solutions, and challenges," *IEEE Access*, vol. 8, pp. 32031–32053, 2020.
[14] S. Azizi, F. Khosroabadi and M. Shojafar, "A priority-based service placement policy for fog-cloud computing systems," *Computational Methods for Differential Equations*, vol. 7, no. 4, pp. 521–534, 2019.
[15] R. Mahmud, K. Ramamohanarao and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Transactions on Internet Technology*, vol. 19, no. 1, pp. 1–21, 2018.
[16] O. Skarlat, M. Nardelli, S. Schulte, M. Borkowski and P. Leitner, "Optimized IoT service placement in the fog," *Service Oriented Computing and Applications*, vol. 11, no. 4, pp. 427–443, 2017.
[17] M. Q. Tran, D. T. Nguyen, V. A. Le, D. H. Nguyen and T. V. Pham, "Task placement on fog computing made efficient for IoT application provision," *Wireless Communications and Mobile Computing*, vol. 2019, pp. 1–17, 2019.
[18] B. K. GJ, "Internet of things (IoT) and cloud computing based persistent vegetative state patient monitoring system: A remote assessment and management," in *Proc. CTEMS*, Belgaum, India, pp. 301–305, 2018.
[19] S. Shukla, M. F. Hassan, M. K. Khan, L. T. Jung and A. Awang, "An analytical model to minimize the latency in healthcare internet-of-things in fog computing environment," *PLOS One*, vol. 14, no. 11, pp. e0224934, 2019.
[20] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing paradigm," in *Proc. SINSM*, Lisbon, Portugal, pp. 1222–1228, 2017.
[21] T. C. S. Xavier, I. L. Santos, F. C. Delicato, P. F. Pires, M. P. Alves *et al.*, "Collaborative resource allocation for cloud of things systems," *Journal of Network and Computer Applications*, vol. 159, pp. 102592, 2020.
[22] T. Wang, Y. Liang, W. Jia, M. Arif, A. Liu *et al.*, "Coupling resource management based on fog computing in smart city systems," *Journal of Network and Computer Applications*, vol. 135, pp. 11–19, 2019.
[23] M. D. Gavaber and A. Rajabzadeh, "MFP: An approach to delay and energy-efficient module placement in IoT applications based on multi-fog," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, pp. 1–17, 2020.

[24] U. Arora and N. Singh, "IoT application module placement in heterogeneous fog-cloud infrastructure," *International Journal of Information Technology*, vol. 13, pp. 1–8, 2021.

[25] F. Faticanti, F. D. Pellegrini, D. Siracusa, D. Santoro and S. Cretti, "Throughput-aware partitioning and placement of applications in fog computing," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2436–2450, 2020.

[26] Q. Shaheen, M. Shiraz, M. U. Hashmi, D. Mahmood, Z. Zhiyu *et al.*, "A lightweight location-aware fog framework for QoS in internet of things paradigm," *Mobile Information Systems*, vol. 2020, pp. 1–15, 2020.

[27] J. P. Martin, A. Kandasamy and K. Chandrasekaran, "Mobility aware autonomic approach for the migration of application modules in fog computing environment," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, pp. 1–20, 2020.

[28] H. Gupta, A. V. Dastjerdi, S. K. Ghosh and R. Buyya, "iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments," *Software: Practice and Experience*, vol. 47, no. 9, pp. 1275–1296, 2017.